Université de Montréal

Finer grained evaluation methods for better understanding of deep neural network representations

par

Florian Bordes

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée en vue de l'obtention du grade de
Philosophiæ Doctor (Ph.D.)
en informatique

Août, 2023

Université de Montréal

Faculté des arts et des sciences

Cette thèse intitulée

Finer grained evaluation methods for better understanding of deep neural network representations

présentée par

Florian Bordes

a été évaluée par un jury composé des personnes suivantes :

Aishwarya Agrawal
(président-rapporteur)

Pascal Vincent
(directeur de recherche)

Christopher Pal
(membre du jury)

Yannis Kalantidis
(examinateur externe)

Marine Carrasco
(représentant du doyen de la FESP)

Résumé

Établir des méthodes d'évaluation pour les systèmes d'intelligence artificielle (IA) est une étape importante pour précisément connaître leurs limites et ainsi prévenir les dommages qu'ils pourraient causer et savoir quels aspects devraient être améliorés. Cela nécessite d'être en mesure de dresser des portraits précis des limitations associées à un système d'IA donné. Cela demande l'accès à des outils et des principes fiables, transparent, à jour et faciles à utiliser. Malheureusement, la plupart des méthodes d'évaluation utilisées à ce jour ont un retard significatif par rapport aux performances toujours croissantes des réseaux de neurones artificiels. Dans cette thèse par articles, je présente des méthodes et des principes d'évaluation plus rigoureux pour obtenir une meilleur compréhension des réseaux de neurones et de leurs limitations.

Dans le premier article, je présente Representation Conditional Diffusion Model (RCDM), une méthode d'évaluation à l'état de l'art qui permet, à partir d'une représentation donnée – par exemple les activations d'une couche donnée d'un réseau de neurones artificiels – de générer une image. En utilisant les dernières avancées dans la génération d'images, RCDM permet aux chercheur · euse · s de visualiser l'information contenue à l'intérieur d'une représentation. Dans le deuxième article, j'introduis la régularisation par Guillotine qui est une technique bien connue dans la littérature sur l'apprentissage par transfert mais qui se présente différemment dans la littérature sur l'auto-apprentissage. Pour améliorer la généralisation à travers différentes tâches, on montre qu'il est important d'évaluer un modèle en coupant un certain nombre de couches. Dans le troisième article, j'introduis le score DéjaVu qui quantifie à quel point un réseau de neurones a mémorisé les données d'entraînement. Ce score utilise une petite partie d'une image d'entraînement puis évalue quelles informations il est possible d'inférer à propos du reste de l'image. Dans le dernier article, je présente les jeux de données photo-réalistes PUG (*Photorealistic Unreal Graphics*) que nous avons développés. Au contraire de données réelles, pour lesquelles générer des annotations est un processus coûteux, l'utilisation de données synthétiques offre un contrôle total sur la scène générée et sur les annotations. On utilise un moteur de jeux vidéo qui permet la synthèse d'images photo-réalistes de haute

qualité, afin d'évaluer la robustesse d'un réseau de neurones pré-entraîné, ceci sans avoir besoin d'adapter ce réseau avec un entraînement additionnel.

Mots cles: Apprentissage profond, Évaluation, Mémorisation, Apprentissage Auto-Supervisé, Données synthétiques

Abstract

Carefully designing benchmarks to evaluate the safety of Artificial Intelligent (AI) agents is a much-needed step to precisely know the limits of their capabilities and thus prevent potential damages they could cause if used beyond these limits. Researchers and engineers should be able to draw precise pictures of the failure modes of a given AI system and find ways to mitigate them. Drawing such portraits requires reliable tools and principles that are transparent, up-to-date, and easy to use by practitioners. Unfortunately, most of the benchmark tools used in research are often outdated and quickly fall behind the fast pace of improvement of the capabilities of deep neural networks. In this thesis by article, I focus on establishing more fine-grained evaluation methods and principles to gain a better understanding of deep neural networks and their limitations.

In the first article, I present Representation Conditional Diffusion Model (RCDM), a state-of-the-art visualization method that can map any deep neural network representation to the image space. Using the latest advances in generative modeling, RCDM sheds light on what is learned by deep neural networks by allowing practitioners to visualize the richness of a given representation. In the second article, I (re)introduce Guillotine Regularization (GR) – a trick that has been used for a long time in transfer learning – from a novel understanding and viewpoint grounded in the self-supervised learning outlook. We show that evaluating a model by removing its last layers is important to ensure better generalization across different downstream tasks. In the third article, I introduce the DejaVu score which quantifies how much models are memorizing their training data. This score relies on leveraging partial information from a given image such as a crop, and evaluates how much information one can retrieve about the entire image based on only this partial content. In the last article, I introduce the Photorealistic Unreal Graphics (PUG) datasets and benchmarks. In contrast to real data for which getting annotations is often a costly and long process, synthetic data offers complete control of the elements in the scene and labeling. In this work, we leverage a powerful game engine that produces high-quality and photorealistic images to evaluate the robustness of pre-trained neural networks without additional finetuning.

Keywords: Deep Learning, Evaluation, Memorization, Benchmarks, Self-Supervised Learning, Synthetic Data

Contents

List of tables

List of figures

Liste des sigles et des abréviations

AI	Artificial Intelligence
MSE	Mean Square Error
DNN	Deep Neural Networks
(D)AE	(Denoising) Auto-Encoders
SSL	Self-Supervised Learning
OOD	Out-Of-Distribution
GAN	Generative Adversarial Networks
RCDM	Representation Conditional Diffusion Model
PUG	Photorealistic Unreal Graphics
VLM	Vision Language Model

Acknowledgements

This thesis would not have been possible without the support of my advisor Pascal Vincent. I will always be grateful to Pascal for taking me as his student during the winter of 2016. From that point on, it has been 7 years of learning and fruitful collaborations. Thank you so much Pascal for the choice you made that day, you have profoundly changed my life.

I also want to acknowledge Tess Berthier and Lisa Di Jorio for being very supportive colleagues while I was interning at Imagia. I also want to acknowledge everyone at the Montreal Meta office. Thank you for taking me as a visiting student and for providing everything that I needed for my work. Thank you Nicolas Ballas, Mido Assran, Diane Bouchacourt, Quentin Duval, Michal Drozdzal, Adriana Romero Soriano, Pascal Vincent, Mike Rabbat and Joelle Pineau for making the office a so great place to work. Thanks to all my (past and present) fellow part-time students for all the incredible discussions: Samuel Lavoie, Jonhatan Labensold, Maxime Wabartha, Mohammad Pezeshki, Reyhane Askari, Mattie Tesfaldet, Arantxa Casanova, Lluis Castrejon. Thank you Glorietta for all the delicious meals!

Thanks to my co-authors for the incredible works we were able to publish: Adriana Romero Soriano, Adrien Bardes, Amir Bar, Amir Globerson, Andrew Gordon Wilson, Ari Morcos, Armand Joulin, Assaf Shocher, Avi Schwarzschild, Casey Meehan, Chuan Guo, Diane Bouchacourt, Gregoire Mialon, Hamed Pirsiavash, Ishan Misra, Jonas Geiping, Kamalika Chaudhuri, Lisa Di Jorio, Mahmoud Assran, Mark Ibrahim, Mathilde Caron, Micah Goldblum, Michal Drozdzal, Mike Rabbat, Mohammad Pezeshki, Nicolas Ballas, Pascal Vincent, Pierre Fernandez, Piotr Bojanowski, Quentin Duval, Quentin Garrido, Randall Balestriero, Reyhane Askari, Samuel Lavoie, Shashank Shekhar, Tess Berthier, Timothée Lesort, Tom Goldstein, Trevor Darrell, Vlad Sobal, Yann LeCun, Yoshua Bengio, Yuandong Tian.

Thanks to the Pascalians: Thomas George, Tom Bosc, Hugo Bérard, Ahmed Touati, César Laurent, Alexandre de Brébisson, Sina Honari, Vincent Michalski, Xavier Bouthilier.

A special thanks to Randall Balestriero whose support has been essential in giving me the confidence I needed as a researcher. I will always be grateful to Randall for supporting the ideas I had and for pushing me to believe in them. Another thanks to Ari S. Morcos who always pushed me to get more ambitious about my work! Thank you Yoshua Bengio and Anne-Catherine Sabas for all the stimulating debates and for your support.

I am profoundly thankful to my beloved wife Tamara Nguyen for her patience during these long and difficult years. I will always be grateful for the infinite support and love I receive. Thank you for always finding the right words when I felt discouraged. I am very lucky to have you as my life partner. I also want to emphasize the contribution of our pets: Jules – a light gray pug who has been a source of inspiration to many things in this thesis – Martha – a tortoiseshell cat who has been a great desk companion – and Louise – probably the smallest chihuahua I ever met who is a source of infinite joy. I am also thankful to my family who despite being far away, has always supported the choices I made.

A PhD can be a very long and difficult journey, there have been many struggling times in which I felt my work was going nowhere. However, with perseverance and the support of everyone I was able to finally get through it. Today, I am very proud to present this thesis.

Introduction

The concept of Artificial Intelligence (AI) has long been a source of inspiration throughout human history. We can find early references to artificial beings endowed with intelligence in Greek mythology. Talos, created by the blacksmith god Hephaestus was made of bronze and was tasked to protect the island of Crete and her queen Europa. Similarly, Hephaestus crafted two guardian dogs for Alcinous, king of the Phaeacians. In contrast with automatons designed to follow a sequence of predetermined instructions, artificial beings like Talos or Alcinous's dogs evolved in complex environments with uncertainty about the sequence of actions to follow. The artificial dogs should be alert about potential invaders, which requires memories of who is used to being in the house. If Alcinous is friendly with a stranger, the dogs should not bark or attack this person. Recognizing situations, meeting new people, and adapting to them, requires a crucial ability which is **learning**. However, even if Hephaestus creatures could learn, they were not perfect. Talos, the protector from Crete, had a weakness, a bronze nail that contained his vital fluid. Medea, a sorceress blocked by Talos on her way to Crete and well aware of Talos' weakness, told him she would make him immortal if he removed his bronze nail. Being convinced by this promise, Talos removed his nail, which destroyed him. Notwithstanding his extraordinary nature, Talos got defeated by mere deceiving words. Even if the ancient Greeks only dreamt about AI, these myths shed some light on a major issue encountered by sophisticated modern systems: with the ability to learn by interacting with the world comes the risk that the system gets deceived by this world into a behavior that was not expected. This calls for the need to create systems that are safe and **robust** to unusual situations when deployed.

Unlike the god Hephaestus who could easily infuse intelligence to bronze statues, modern AI systems rely on a set of computing techniques called deep learning. Scientists inspired by the underlying mechanisms of the brain have tried to distill these mechanisms into mathematical formulas and algorithms. Such modern artificial systems are built upon the notion of a large number of connected artificial neurons: a neural network. When many layers of connected neurons are linked sequentially, they are referred to as deep neural networks. Like Alcinous's dogs, deep neural networks must be able to learn to be qualified as intelligent

systems: they need **deep learning**. However, artificial beings or deep neural networks cannot learn from nothing. They are crafted with a task in mind; it can be as specific as alerting when someone enters the house to as general as holding a friendly conversation. Learning to carry out such a task is often referred to as *training* the deep neural network. Training can be performed either using direct human feedback (using knowledge about the answer that a human would give), which is called *supervised learning*, or indirect feedback (is the answer likely given a predefined series of rules about the world?) with *unsupervised* or *self-supervised* learning (SSL). Since we do not want to create an AI system that a user can easily trick into a behavior that is not desirable, it is crucial, after training, to evaluate the system's robustness to guarantee its safe deployment.

Carefully evaluating and designing benchmarks to measure AI systems' robustness is at this thesis's foundation. As teachers test their students to measure how much they learned, we must evaluate how much deep neural networks have learned. However, we must be careful when designing such tests. Like a student who might achieve a perfect score on a test by merely memorizing the correct answers without understanding the fundamentals, an AI system can also give correct answers on basic tests using rote memorization but perform poorly on more challenging tests. In addition to providing a view of the system's failure modes, better evaluation methods could also give us a better understanding of what precisely they learned and how. Are they *cheating* to solve the task? Did they memorize what they learned? What happens if we try to trick them into giving an incorrect answer? The main driving force behind the following articles is providing better tools to researchers to answer these questions.

The first part of this thesis introduces **RCDM (Representation Conditional Diffusion Model): a new tool used to understand better what modern AI systems learn**. Traditionally, assessing what is learned by deep neural networks is mostly measured through numerical performances on the task of interest. When training them for object recognition, the standard evaluation benchmark is to compute the accuracy on an annotated set of data not seen during training (to evaluate how robust this network is to new data or environments). However, the downside of this kind of evaluation is that it only assesses if the neural network is *classifying* correctly, without uncovering the reasoning behind the prediction. This network can learn to recognize birds by using only the background information (birds will often have a sky background) instead of what uniquely characterizes a bird (peak, wings..). In that instance, the network would be *cheating*, and the evaluation benchmark might not be able to detect it unless it had been carefully designed based on such a-priori expected misbehavior. To circumvent these limitations, we propose using a rich qualitative evaluation method to visualize directly what a deep neural network has learned. To this effect, we introduce RCDM, based on the latest advances in image generation.

RCDM can stochastically map any deep neural network layer representation to the image space. This allows us to visualize precisely what information is contained in this representation.

In the second part, we **analyze what is learned by neural networks at different intermediate levels of representations by using Guillotine Regularization**. Nowadays, deep neural networks are built with dozens or hundreds of layers, and each of those layers could contain thousands or millions of artificial neurons. There is a habit in the research community to always leverage the last layer to solve a given downstream task. Our second contribution is challenging this habit by showing that different layers contain different information about a given input. Thus, some intermediate representations might be better suited for some tasks. This is especially true with Self-Supervised Learning approaches, for which performance across different downstream tasks can vary significantly depending on which representation layer is used.

In the third part, I introduce **DéjàVu: a score to quantify memorization in deep neural networks**. Since a student could learn by rote memorization to get good grades, a neural network could do the same to solve the training task. This kind of memorization is an especially sensitive topic with deep neural networks since it could entail risks of privacy violation. An attacker could try to reconstruct the training data, which might induce significant risks when private or proprietary data is used for training. In this third contribution, we highlight how much self-supervised models rely on memorization and which parameters increase or decrease the degree of memorization of the training data. By introducing a new metric, called the DéjàVu score, we give SSL researchers and practitioners a new tool to analyze how much a given model memorizes its training data.

Lastly, we introduce the **PUG (Photorealistic Unreal Graphics) datasets and benchmarks**. Most of the datasets and benchmarks used by the research community quickly become outdated, falling behind the fast pace of improvement of the capabilities of deep neural networks. Most synthetic datasets are often small-scale and unrealistic, while real datasets lack fine-grained control and annotations. Since transparency on a model's limitations is crucial to ensure their safe deployment, it is fundamental to create benchmarks to properly assess the limitations and risks associated with a given model. This work is inspired by the simulators used in aviation to evaluate how well pilots can react to extremely rare but possible failure cases. Using a simulator allows pilots to sharpen their skills in case of unlikely events. Similarly, using synthetic yet realistic-looking data, we can easily evaluate how much models are robust to unlikely scenes or events. This last contribution introduces the PUG datasets created using Unreal Engine – a powerful video game engine. By leveraging the latest photorealistic graphics, we can create virtual environments in which

we can asses the robustness of modern systems. Equipped with PUG, we create challenging configurations under which current neural networks are easily shown to fail.

The four articles presented in this thesis aim to give practitioners and researchers additional guidance and tools to evaluate deep neural networks better. Much in these articles are centered on deep neural networks that belong to the Self-Supervised learning family. Unlike traditional supervised learning, for which the training objective is typically aligned with their intended usage, Self-Supervised Learning methods solve a more generic objective to learn features that can be used across different downstream tasks. Consequently, having a way to correctly evaluate these generic models before releasing them is essential to draw a better picture of their strengths and weaknesses. However, the guidance and tools presented in these articles can be generalized to other model families. I hope these contributions will be the foundation for crafting more robust, safe, and reliable artificial intelligent systems. This thesis starts with a background chapter in which I introduce the fundamental notions needed to understand the articles. Then, each article is presented with a prologue that provides context and information on its conception and highlights its impact on the research community. Lastly, a conclusion chapter summarizes the contributions and opens up to the remaining research directions and questions to address. The supplementary material specific to each article can be found at the end of the thesis, after the bibliography.

Chapter 1

Background

This chapter presents an overview of the fundamentals needed to understand the articles presented in this thesis. It starts with machine learning concepts that are at the core of modern AI systems. Then, I explain neural networks, their most common forms, and popular training paradigms. Lastly, I review the latest advances in Self-Supervised Learning.

1.1. Machine learning basics

1.1.1. Introduction

Machine learning is an ensemble of techniques that aim to solve specific tasks using data. Tasks could be as simple as drawing a line between two clouds of points or much more complex such as labeling different animal species or playing Go. Labeling species requires understanding of what an animal is and what are the characteristics of a given animal. It leads to learning *abstractions* of concepts such as tail or legs. How to learn better abstractions is an essential question in machine learning research.

1.1.2. Solving tasks

Machine learning algorithms learn to solve a task by optimizing a predefined training *objective*. In a classification task, the algorithm can predict for a given image which category is the most likely to be associated with this image. This first task belongs to a *supervised learning* scenario for which we have category labels, i.e., annotation of the data. This allows machine learning practitioners to evaluate if an algorithm made a mistake by comparing the predicted label to the real label. If the task is to generate new images of cats, the objective is much harder to define. We can ask the algorithm to produce a very close pixel-wise image to a target image or to produce indistinguishable images from real images under the view of a second algorithm. This second task belongs to an *unsupervised learning* scenario for which we have data with no annotations on them. Similarly, we can define a *pretext objective* which

might be as generic as "trying to predict a missing word in a sentence" or "a missing patch in an image". By training on this pretext task, the model might learn representations that are *generic enough* to be used for different downstream tasks (like classification or image segmentation). This third scenario is referred to as Self-Supervised Learning.[1]

1.1.3. Learning

Learning consists mainly in leveraging the training data to find a function (a specific parametrization) that best solves the training objective. In most machine learning algorithms, variables called parameters are used to represent that function. Usually, those parameters are initialized randomly. Then, the algorithm finds the best parameter values that optimize the algorithm's performance in a given task. Finding the optimal parameter values for a given training task is often referred to as the training phase and is usually performed until the task is solved *sufficiently well*.

1.1.4. Evaluating

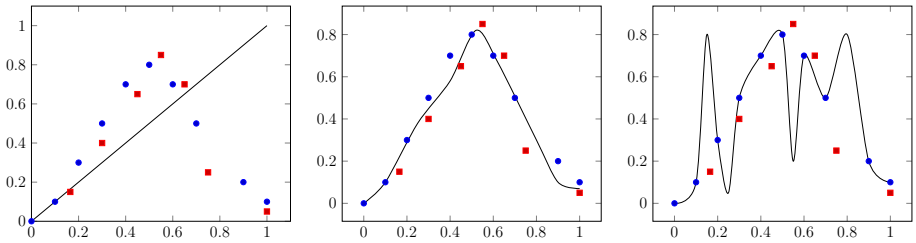
The evaluation of a machine learning algorithm depends on the chosen downstream task. In the case of supervised learning, one usually splits the data into three different sets. The first is the training set the machine learning algorithm uses to learn its parameters. The second one is the validation set, which tracks performances on unseen data to assess whether the algorithm is only memorizing the training set or can generalize on new data. However, tracking performances on a specific set of unseen data can lead to a bias towards this set, especially if we use this information to choose the algorithm. To prevent it, we use a third set called the test set, which the algorithm did not use during the training phase. The performances on this set should give us an unbiased estimate of the algorithm's actual performances.

1.1.5. Capacity, generalization, under and over-fitting

The capacity of a machine learning algorithm can be seen as the expressiveness or representation power it can have. A small capacity implies that the algorithm will be able to solve only simple tasks, whereas an algorithm with a bigger capacity will be able to solve more complex tasks. Usually, the capacity is related to the number of parameters the algorithm can use. We often associate the capacity with the function space: the set of functions the algorithm can learn. If we look at Figure 1a, the degree of the polynomial function is 1, which restricts the function space to the set of linear functions such as $f(x) = ax$ with a

[1]There are many ways to better characterize the differences between Unsupervised and Self-Supervised Learning. For simplicity, we restrict our definition of Unsupervised Learning to learning the data distribution (thus learning a model that can generate new data points). In contrast, we define Self-Supervised Learning as leveraging a pretext task to learn a generic representation.

as the learnable parameter. However, the algorithm tries to learn the blue training data points, which a linear function cannot represent. In that case, the linear function is not fitting the training data points well. Since the expressiveness of the linear function does not allow the algorithm to represent the data correctly, we are in an *under-fitting* scenario. If the polynomial degree becomes higher, as shown in Figure 1b, the algorithm has a bigger capacity and is better at capturing the training data distribution. In addition, the test data points in red are also captured by the learned polynomial. In that instance, the algorithm *generalizes* to the test data points. However, when the polynomial degree is too high, as shown in Figure 1c, the algorithm's capacity becomes too strong to offer good generalization. In that instance, the model learns precisely the position of the training blue data points but fails to learn a function that can represent the test red data points. In that instance, the algorithm is in an *over-fitting* scenario.



(a) Underfitting. **(b)** Good trade-off. **(c)** Overfitting.

Fig. 1. Plots showing how the capacity of a model influences its ability to fit data points. Train points are blue, and test points are red. The polynomial learned by the machine learning algorithm is the black curve. Figure extracted from my master thesis.

1.2. Neural networks

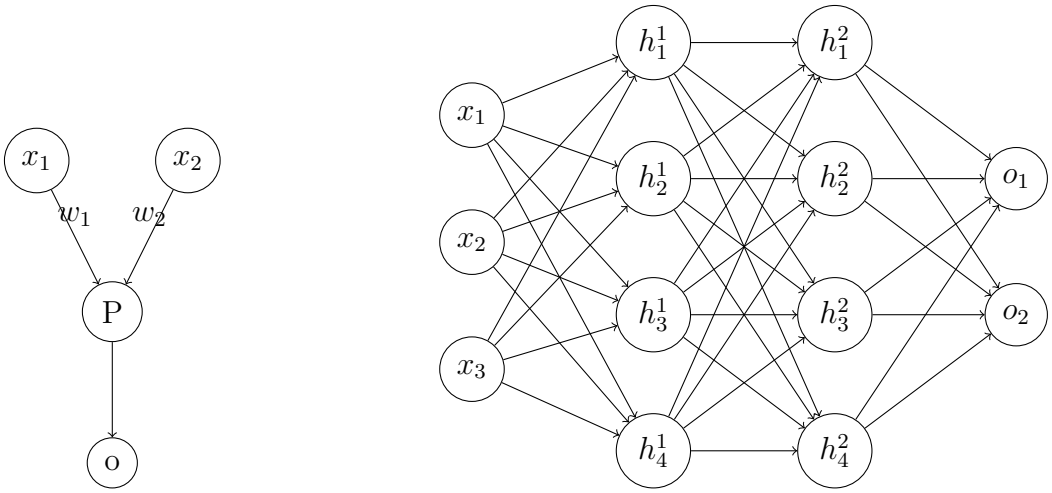
1.2.1. Introduction

Neural networks are a class of machine learning models inspired by neuroscience. Neural networks are composed of artificial units called neurons whose links are called weights. In contrast with other machine learning models, neural networks make the compositionality assumption that complex concepts can be built using simpler elements. Such an assumption has a lot of grounding in neuroscience since our brain can decompose a complex animal like a cat under simpler concepts such as paws, tail, pointy ears, whiskers, and small. If taken individually, each element does not imply they are associated with a cat since they are common across many different animals. However, if all of them are detected in the same animal, we can definitively assume it is a cat. If we see a tiger, our brain can think about how big this cat is compared to our domestic cat (such an association occurs only because cats and tigers share many characteristics). The ability to reuse simple concepts and

associations we learned in the past is a crucial element that allows our brain to generalize to new environments. Learning simple concepts and combining them is one of the main reasons behind the success of neural networks.

1.2.2. Perceptron and MultiLayer Perceptron

Inspired by discoveries about the brain, Rosenblatt invented the perceptron [171], an artificial neuron. The perceptron (Figure 2a) can be seen as a simple linear combination of the inputs with a threshold. The multilayer perceptron (Figure 2b) was introduced to solve non-linear problems like XOR. Instead of using one artificial neuron, the idea is to stack them inside a layer of neurons and combine them with other layers of neurons and non-linear activation functions.



(a) A model of a Perceptron **(b)** A Multilayer Perceptron

Fig. 2. Example of perceptron and multilayer perceptron. The input variable is named x_i whereas the output given by those models is called o_j. The artificial neuron units are presented as h_j^i.

1.2.3. Convolutional Neural Networks

Convolutional neural networks (CNNs) [130] are a specific type of neural network designed for vision tasks. In contrast to fully connected networks that would associate every pixel to each of the neurons, CNNs have grids of neurons that analyze only a subset of the pixel to find some patterns. When having as input an image of a dog (Figure 3), a specific subset of these neurons will learn to look for something like an ear. They will scan the entire image grid of pixels by grid of pixels and send a signal to the bottom layer if they find something that looks like an ear.

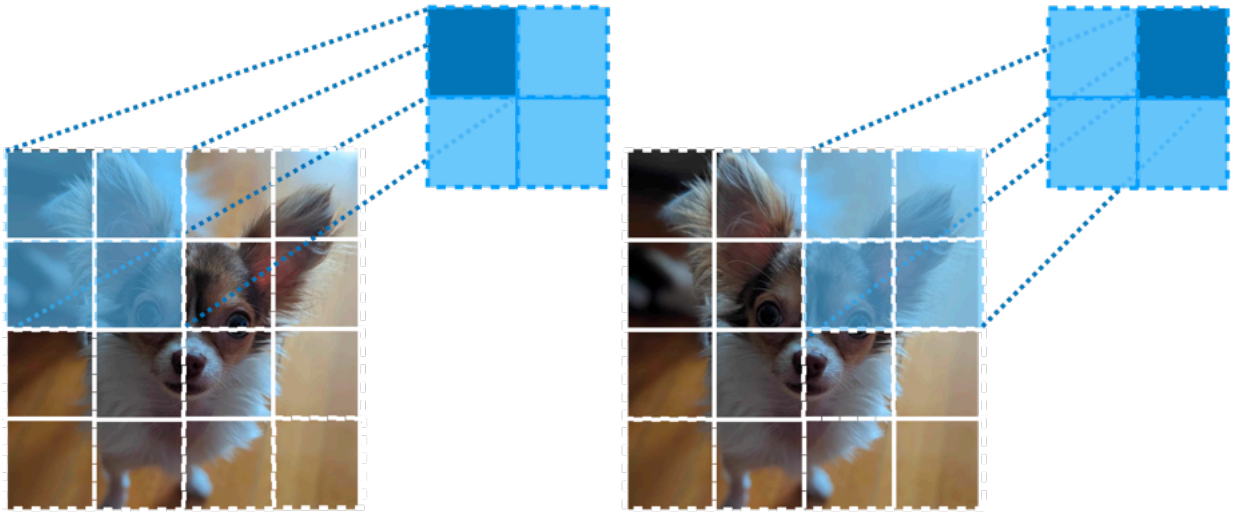


Fig. 3. An example of a convolution operation applied on an image. In contrast to fully connected networks that associate every pixel to each of the neurons, CNNs analyze only a subset of the pixels (in transparent blue) to compute the higher-level activation (in dark blue).

1.2.4. Transformers

Transformers are a specific type of deep neural network architecture introduced by Vaswani et al. [210] initially designed for text applications. One of the main components of the transformer architecture is the attention mechanism introduced by Bahdanau et al. [15]. The text inputs are converted into *tokens*, which then are converted into *embeddings* on which multiple attention heads are deployed with non-linear activation and feed-forward layers. By using the attention heads, transformers can learn to attend to specific part of a given sequence depending on the training task. One of the main advantages of such an approach is the computational cost which is significantly reduced since the network does not need to focus on the irrelevant part of a given sequence.

1.2.5. Cost function, and backpropagation

A neural network, like any machine learning algorithm, should optimize a cost (or loss) function to solve a training task. For a supervised classification task, the cost will be how many errors a model makes in predicting the class labels. Training a machine learning algorithm based on a neural network is done by reducing the number of errors made (minimizing the cost) on the training data. To minimize the cost function, we use optimization techniques based on gradient descent to find the optimal parameters for the neural network. Backpropagation of the gradient [173] is a method to efficiently compute gradients adapted to neural networks.

Based on the chain-rule principle, we compute the gradient at different layers, starting with the last one until the first one.[2].

1.2.6. Autoencoders

Autoencoders are a class of neural networks trained to reconstruct their input through a specific architecture. The intuition behind autoencoders is that a model can learn a meaningful representation of the data by encoding an input into a specific number of dimensions in a latent space. It is important to note that the autoencoder could *cheat* by learning to copy its input. Thus, it is essential to design the autoencoder to avoid that by reducing the dimension of the input data (undercomplete autoencoder) and using non-linear activation functions. An alternative is to design autoencoders that project the data into a higher dimension (overcomplete autoencoders) with strong regularization over the latent or input space. One popular form of regularized autoencoders is **denoising autoencoders (DAE)** [214]. Instead of directly using the training data point as input for the autoencoder, the input becomes a slightly corrupted version of this data training point. Then the DAE learns to reconstruct the original data point from this noisy version. This can done by using the following Mean Squared Error (MSE) objective:

$$J_{MSE}(\theta) = \sum_{x \in D_{train}} \mathbb{E}_{\tilde{x} \sim \mathcal{N}(x, \sigma^2 I)} \left[\frac{1}{2} \left\| \phi(\tilde{x}, \theta) - x \right\|^2 \right]$$

where x is the input, $\tilde{x}$ the slightly corrupted version of x and $\phi(\tilde{x}, \theta)$ the DAE parametrized by θ which takes $\tilde{x}$ as input. In that instance, the autoencoder cannot merely copy its input since the reconstruction error with respect to the clean data point will be high. Figure 4 gives an example of denoising autoencoder. The DAE learns to map the corrupted version of the data points (in blue) toward the data manifold (in green).

1.2.7. Energy-based models and denoising score matching

Energy-based models (EBMs) [131] are a specific framework that predicts one scalar value called energy for each configuration of random variables. The model E_θ parametrized by θ is trained to assign low energy on observed variables and high energy on unobserved ones. Data from the target distribution should have low energy, whereas anything else should have higher energy. EBMs do not require proper normalization but correspond to a Boltzmann distribution (Or Gibbs distribution). We consider input data x with an energy function $E_\theta(x)$ of parameters θ. The corresponding Boltzman distribution density function can be written as:

$$p(x) = \frac{e^{-E_\theta(x)}}{Z_\theta}$$

[2]For a more detailed explanation, please refer to the book of Goodfellow et al. [90]

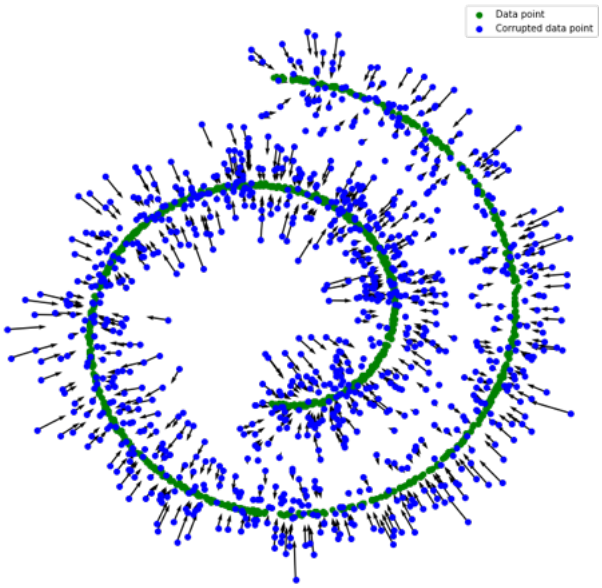


Fig. 4. An example of a denoising autoencoder (DAE) that learns to map noisy versions of data points towards the data manifold. The data points are green, whereas the corrupted ones are blue. The DAE learns to project back the blue point towards the green data points.

with $Z_\theta = \sum_x e^{-E_\theta(x)}$ which is the normalization factor. To estimate the target distribution P_D from which input data x are drawn, we can in principle use the traditional maximum likelihood objective:

$$\arg\min_\theta \mathbb{E}_{x\sim P_D(x)}[-\log P_\theta(x)]$$

whose gradients are:

$$\frac{\partial \mathbb{E}_{x\sim P_D(x)}[-\log P_\theta(x)]}{\partial\theta} = \mathbb{E}_{x^+\sim P_D(x)}\frac{\partial E_\theta(x^+)}{\partial\theta} - \mathbb{E}_{x^-\sim P_\theta(x)}\frac{\partial E_\theta(x^-)}{\partial\theta}$$

However, this requires to get $x^- \sim P_\theta(x)$, which corresponds to sample from the model distribution that can be intractable. A common strategy is to use Markov Chain Monte Carlo (MCMC) methods to simulate the sampling distribution of the EBM. The advantage of such an approach is that there are theoretical guarantees that the chain will converge, but there is a substantial computational cost.

To overcome such training difficulty, a popular workaround is to use score matching [109] to train EBMs. Instead of considering the density itlsef, we use the gradient of the density $\phi(x,\theta)$, called the score, where:

$$\phi(x,\theta) = \frac{\partial \log p(x,\theta)}{\partial x} = \frac{-\partial E_\theta(x) - \partial \log Z_\theta}{\partial x} = \frac{-\partial E_\theta(x)}{\partial x}$$

Score matching eliminates the partition function $\log Z_\theta$ since the score does not depend on x, leading to a zero gradient.

We need to define a score as a target to train a model by score matching. We would like $\phi(x, \theta)$ as close as possible to the score $\frac{\partial \log q(x)}{\partial x}$ of the true (and unknown) data density q. If we knew this term, we could train a model to minimize:

$$J_{SM}(\theta) = \sum_{x \in D_{train}} \frac{1}{2} \left\| \phi(x, \theta) - \frac{\partial \log q(x)}{\partial x} \right\|^2$$

but we don't know the real $\frac{\partial \log q(x)}{\partial x}$.

Since we do not have the real $\frac{\partial \log q(x)}{\partial x}$, the idea behind denoising score matching introduced by Vincent [212] is to add some noise around a data point x to get a new point $\tilde{x}$. The intuition is that if we were to follow the true gradient $\frac{\partial \log q(\tilde{x})}{\partial \tilde{x}}$, we would likely get a gradient that will move $\tilde{x}$ towards x. Thus, we can define a new objective:

$$J_{DSM}(\theta) = \sum_{x \in D_{train}} \mathbb{E}_{\tilde{x} \sim \mathcal{N}(x, \sigma^2 I)} \left[\frac{1}{2} \left\| \phi(x, \theta) - \frac{(x - \tilde{x})}{\sigma^2} \right\|^2 \right]$$

Interestingly, Vincent [212] demonstrated that the denoising score matching criterion $J_{DSM}(\theta)$ is equivalent to the Mean Squared Error Loss $J_{MSE}(\theta)$ that is used when training a denoising autoencoder.

The DSM objective has been widely used to train the generative models, which are presented in the next section.

1.2.8. Diffusion/Infusion models

Diffusion models are a family of generative models inspired by non-equilibrium statistical physics. They were first introduced and described by Sohl-Dickstein et al. [191] as composed of a diffusion process (which consists of converting a complex distribution into a simple and tractable distribution by iteratively corrupting it) and the corresponding reversed diffusion process. If we modelize distribution of images, then the diffusion process will corrupt iteratively these images (using Gaussian noise, for example) such that after a given number of iterations, the resulting corrupted images become purely Gaussian. The diffusion model will learn to reverse each intermediate step such that from pure Gaussian noise, the network should gradually generate the target images back. To learn the exact reverse process, the diffusion had to be infinitesimal by construction which requires performing thousands of tiny reconstruction steps (which leads to important computational costs). In this first generation of diffusion models, the training loss consisted of optimizing a lower bound over the log-likelihood. Diffusion models are akin to denoising autoencoders with the difference that they have iterative components and use a more complex process to corrupt the data. The diffusion models introduced by Sohl-Dickstein et al. [191] worked well on simple data

distributions however, they were not able to generate compelling images.

Fig. 5. An example of an infusion chain by Bordes et al. [28]. Starting from noise, we gradually infuse very few target pixels into the sampling chain. By doing so, the network learns to construct back the image gradually. At inference, we remove the infusion steps to sample from the model.

In a following work, Bordes et al. [28][3] demonstrated that diffusion models could be simplified and improved in such a way that they could outperform all other generative models that were available at the time of publication. Instead of optimizing a lower bound over the log-likelihood, we can directly predict the true target x at each time step, corresponding to a simple denoising score matching criteria. Instead of learning the exact diffusion reversed process (and their thousand of tiny steps), we introduced another heuristic coined as *infusion* that randomly replaces pixels with true target image pixels during the iterative denoising training process. This approach had the main advantage of requiring only a dozen steps for sampling from the model. In this work, we also discovered that we could significantly improve the quality of the sampled images by conditioning the model with the current denoising timestep.

Despite this progress, the research scene at that time was mostly focused on generative adversarial networks (GANs [88]). Diffusion models resurfaced some years later with the work of Song and Ermon [193], who rediscovered independently that a denoising score matching criteria is well suited to train these models by making a theoretical connection with Langevin dynamics. Similarly to our work, they emphasized the importance of using a conditional network that considers the current timestep or noise level applied to the input. The interest

[3]This work was the subject of my master thesis: Learning to sample from noise with deep generative models.

in diffusion models began to increase significantly after the work of Ho et al. [105], who successfully scaled diffusion model generation to high-resolution images. Then, Dhariwal and Nichol [60] demonstrated that diffusion models can achieve state of the art in image generation.

1.3. Self-supervised Learning

Note: Most of this section either reproduces or summarizes the following work I coauthored: A Cookbook of Self-Supervised Learning [17]. Even if this paper is not part of the articles presented in this thesis, it constitutes a nice introduction to self-supervised learning, which I invite the reader to read.

Self-Supervised Learning is a machine learning paradigm in which we do not use human-annotated data to train the model. In contrast with traditional generative unsupervised learning, SSL practitioners and researchers define an unlabelled pretext training task for the model to learn a representation that is *generic enough* to be able to generalize across a wide range of modalities. Pretext tasks used in self-supervised methods are often very simple: from predicting missing instances in sentences to predicting a specific image's patch content using another patch from the same image. Another successful pretext task of SSL is to leverage artificial data augmentations to create invariances which were shown to be beneficial for learning semantic information about images.

1.3.1. A brief history of Self-supervised Learning through pretext tasks

Contemporary methods build upon the knowledge we gained from early experiments. While many of the specific methods have fallen out of mainstream use because they no longer provide state-of-the-art performance on benchmark problems, the ideas from these papers form the foundation for many modern methods. For example, the core objective of restoring missing or distorted parts of an input or contrasting two views of the same image forms the foundation for modern SSL methods. Early progress in SSL focused on the development of methods that fell into the following (sometimes overlapping) categories:

1. Information restoration: Many methods have been developed that mask or remove something from an image and then train a neural network to restore the missing information. Colorization-based SSL methods convert an image to grayscale and then train a network to predict the original RGB values [242, 128, 216]. Because colorization requires understanding object semantics and boundaries, colorization was demonstrated as an early SSL method for object segmentation. The most straightforward application of information restoration

is to mask, aka remove, a portion of an image and then train a network to inpaint the missing pixel values [164]. This idea evolved into masked auto-encoding methods [99], in which the masked region is a union of image patches that can be predicted using a transformer.

2. Learning spatial context: This category of methods trains a model to understand objects' relative positions and orientations within a scene. RotNet [85] masks the direction of gravity by applying a random rotation and then asks the model to predict the rotation. Doersch et al. [62] is one of the first SSL methods that simply predicts the relative location of two randomly sampled patches in an image. This strategy was superseded by "jigsaw" methods [164, 159] that break an image into an array of disjoint patches and predict the relative location of each.

3. Grouping similar images together: One can learn rich features by grouping semantically similar images together. K-means clustering is one of the most widely used methods from classical machine learning. A number of studies have adapted k-means to perform SSL with neural networks. Deep clustering alternates between assigning labels to images by performing k-means in the feature space and updating the model to respect these assigned class labels [39].

4. Layerwise pretraining: An early influential SSL method is greedy layer-wise pretraining [25], in which deep network layers are trained one at a time using an autoencoder loss. Later advancements improved the representation learning ability of auto-encoders, including denoising autoencoders [214], cross-channel prediction [243], and deep canonically correlated autoencoders [218]. Nonetheless, it was ultimately found that representation transferability is better when the auto-encoder is asked to restore a missing part of its input, resulting in the "information restoration" category of SSL methods.

5. Multi-view invariance: Many modern SSL methods, especially those I am focusing on in this thesis, use contrastive learning to create feature representations invariant to simple transforms. The idea of contrastive learning is to encourage a model to represent two augmented versions of an input similarly. Several methods led the change in this direction by enforcing invariance in various ways before contrastive learning was widely adopted.

One of the most popular frameworks for learning from unlabeled data is to use a weakly trained network to apply pseudo labels to images and then train using these labels in a standard supervised fashion [132]. This approach was later improved by enforcing invariance to transformations and training a network to maximize the mutual information between the representations of an image under different views [14]. These augmentation-based methods

formed a bridge between the older practices described above and the contemporary methods that are the focus of this thesis.

1.3.2. A brief overview of some successful SSL methods

Many SSL methods have been introduced in the last few years. In this thesis, we will focus only on the ones discussed below.

Contrastive based methods (SimCLR) The first one is named *SimCLR* and belongs to the family of contrastive SSL methods. Its pretext task is to learn multiview invariances defined by a set of handcrafted data augmentations. However, this learning objective is insufficient since a DNN could cheat by assigning the same representation to all images in the training set[4]. To avoid such shortcomings, SimCLR introduces the use of *negative* examples. Inspired by the training of energy-based models, SimCLR minimizes the energy of two views of a given image while maximizing the energy of every other pair of images in a given mini-batch. Many have believed using a large batch size, thus having enough negative pair of images, was needed for SimCLR to perform well over a dataset like ImageNet [56]. However, using small batch sizes for training is fine as long as the learning rate is chosen accordingly, as demonstrated by Bordes et al. [32].

The Canonical Correlation Analysis Family Many works have suggested removing the negative examples sampling in SimCLR to focus more on a geometry-based regularization using the Canonical Correlation Framework (CCA) [107]. The high-level goal of CCA is to infer the relationship between two variables by analyzing their cross-covariance matrices. These ideas were extended to deep learning in Deep Canonically Correlated Autoencoders (DCCAE) – an autoencoder regularized via CCA. Hsieh [108] and Andrew et al. [4] introduce the objective of jointly learning parameters for two networks, f_1, f_2, such that their outputs are maximally correlated. From these origins stems SSL methods such as VICReg [21], Barlow Twins [236], and SWAV [41]. **VICReg**, the most recent among these methods, balances three objectives based on co-variance matrices of representations from two views: variance, invariance, and co-variance shown in Figure 6.

The Self-Distillation family Another type of SSL method belongs to the Self-Distillation family. Instead of having an explicit term to avoid collapse of the representations, some techniques like Byol[93] or Dino[43] rely on a teacher network (also coined often as a momentum encoder) for which gradients are not computed. Then a student network is trained to predict the representation that the teacher network predicts. The teacher network

[4]this scenario is often referred to as the *collapse* of the SSL representations

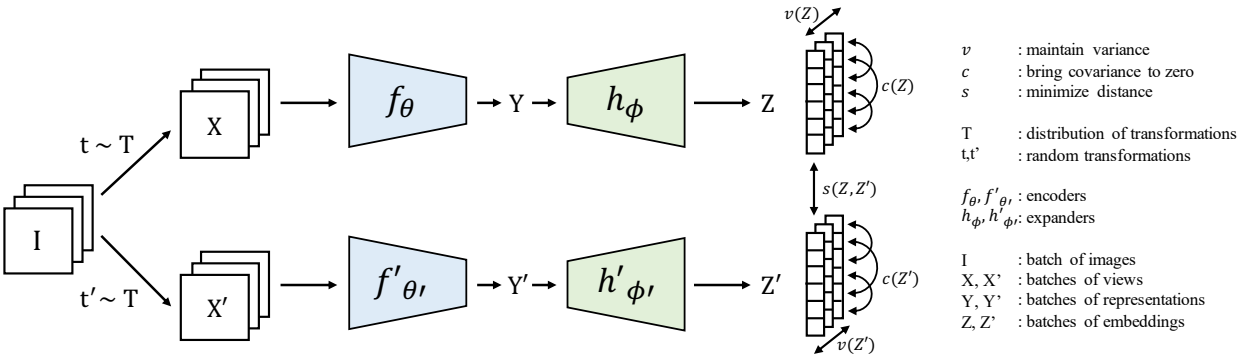


Fig. 6. VICReg: penalizes variance, invariance, and co-variance terms to learn representations from unlabeled data. Regularizing the variance along each dimension of the representation prevents collapse, the invariance ensures two views are encoded similarly, and the co-variance encourages different dimensions of the representation to capture different features. Figure from Balestriero et al. [17].

is updated as an exponential moving average built with the student model gradients update.

Masked Image Modelling (MIM) BERT [59] shook up the natural language processing world by replacing text tokens input to a transformer language model with learnable mask tokens and teaching the model to recover the original text. Inspired by its success, several works mask out portions of an image and train a model to inpaint them. This strategy is known as masked image modeling (MIM). Inspired by BERT, Dosovitskiy et al. [64] exploits the vision transformer architecture by masking out patch tokens and replacing them with learned mask tokens. To streamline MIM pre-training, two concurrent works [99, 227] propose simplified algorithms, masked autoencoders (MAE) and SimMIM, respectively, which directly reconstruct masked image patches. Lastly, Assran et al. [10] demonstrated that instead of using a decoder to reconstruct a patched image, one could learn a predictor trained to predict the representation of a given masked patch.

1.3.3. The critical ingredients for a successful SSL recipe

As presented in the previous section, there are many training pretext tasks and training losses in Self-Supervised learning. However, most of those methods share a standard recipe, allowing them to become serious competitors of traditional supervised approaches. The first ingredient was to use strong data augmentations to generate different views of a given image. Most of them were intuitively designed as a natural methods for learning invariances that could improve classification performance. Color-based augmentations might push the network to learn more information about the shape of the objects, while random cropping operation forces learning semantic information by associating different parts of an object. In Figure 7, we show how various data augmentations impact the performance accuracy of two

different SSL methods trained on ImageNet. Using cropping alone is *not enough* to learn good representations for a classification task, whereas adding color transformation such as Grayscale or ColorJitter significantly improves the performances of the SSL methods.

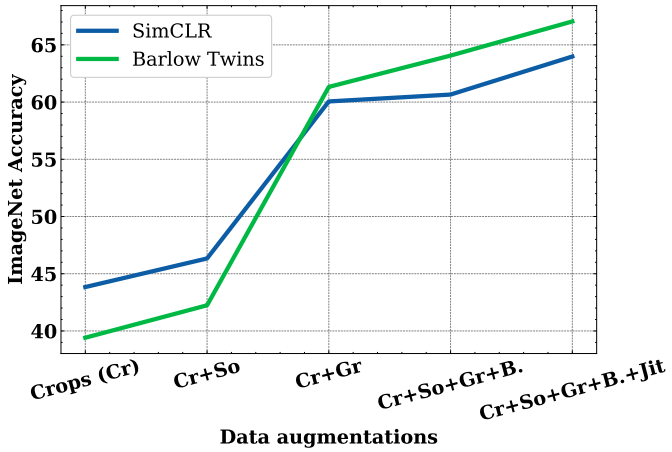


Fig. 7. Detailed impact of the data augmentations used during SimCLR and Barlow Twins training on the ImageNet validation accuracy. *So* corresponds to a Solarization transformation applied with a 20% probability, *Gray* corresponds to a Grayscale operation that is also applied with a 20% probability, *B.* corresponds to a Gaussian blur applied 100%of the time and *Jit* is the ColorJitter operation with 80% probability. In this Figure, we clearly see that adding grayscaling has the most significant impact on ImageNet accuracy. Figure from Bordes et al. [32].

The second key ingredient for a successful SSL recipe was introducing a projection head during training. This projection head is often a 2- or 3-layer multi-layer perceptron with non-linear activation, which is thrown away once the training is done. The primary justification for using such a layer was given in the SimCLR paper [238], which states that the invariances learned by the SSL criteria are too strong for a classification task. However, the use of a projector head for learning SSL representations can be counterintuitive since one promise of Self-Supervised learning is to build representations that are *generic enough* to solve different downstream tasks. Learning correct invariances could allow SSL researchers and practitioners to carefully structure an SSL representation so that a specific part of the SSL representation might be well suited for classification while another part of the representation might be more suited for segmentation tasks. However, by cutting the projector head, we lose most of the invariances learned. Understanding the role of the projector in SSL is a crucial contribution to this thesis which is discussed in depth in Chapter 3.

1.4. Vision Language Models (VLMs)

Vision language models are another model class that learns a text and image representation. One popular way to train such a model is by using contrastive training[165], which is similar

to the contrastive SSL SimCLR method with the exception that the image embedding is contrasted with text embedding (instead of another image embedding). As shown in Figure 8, the VLM often comprises an image encoder that produces an image embedding and a text encoder that produces text embedding. Then the model is trained to minimize the distance between a given image embedding and its corresponding text embedding while increasing the distance with all other text embeddings that do not describe the image.

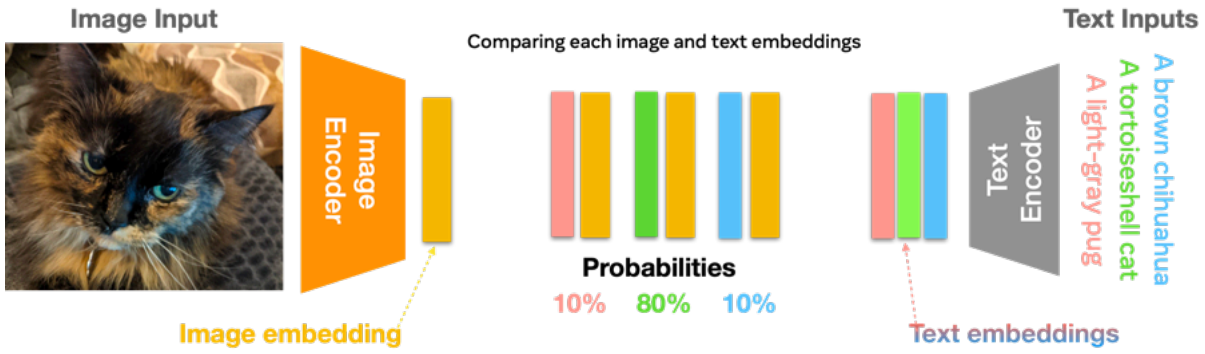


Fig. 8. An illustration of Vision Language Models. They are often composed of an image encoder which produces an image embedding, and a text encoder which produces a text embedding. The model learns to assign the highest probability to the text that describes the image while assigning a lower probability to the text that does not describe the image.

1.5. Evaluations of deep neural network representations

Note: This section is only a glimpse and high-level overview of the evaluation methods that existed before the articles presented in this thesis. A more in-depth literature review is available in each of the articles.

A critical issue with neural networks is that they can be considered as *black boxes*. In contrast to simpler and interpretable ruled-based systems, there is no straightforward verbalizable way to know why a given neural network has associated a specific output with a given input. Knowing that most modern architectures leveraged millions or billions of artificial neurons, it is impossible to precisely know what made the network perform a given computation between an input and output. This lack of interpretable information concerning how a neural network made a decision is an issue for deployment since it is impossible to know in advance if such a model is reliable. The network could have learned to cheat to solve the task of interest by relying on a spurious feature, such as the sky, to classify birds from cars. Without data containing flying cars, it would be impossible to use only the classification accuracy to know whether the network has cheated. In consequence, interpretability is an important and very active area of research.

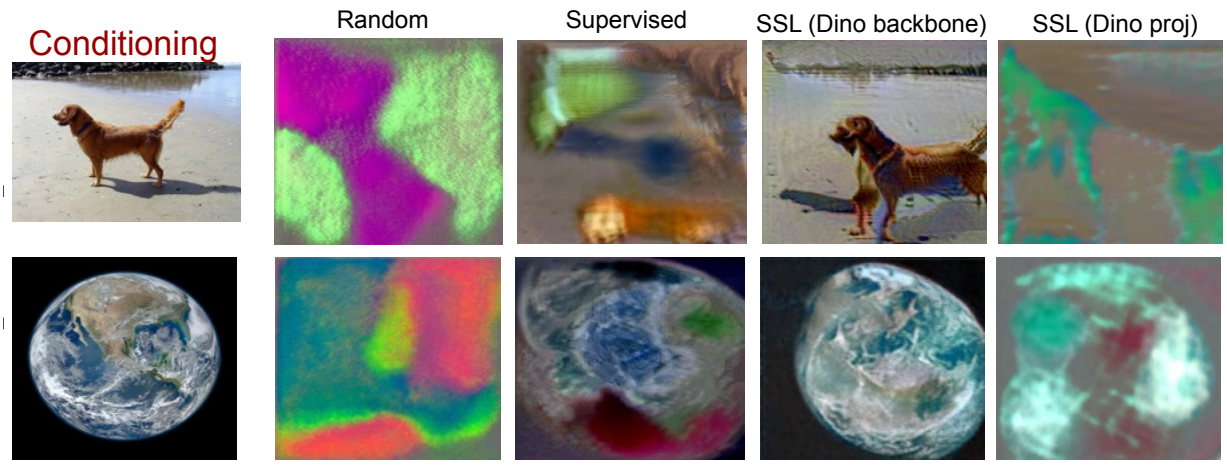


Fig. 9. Interpretability with Deep Image Prior (DIP). In this figure, we took the representation of two images (dog and earth) extracted with different models (random, supervised, SSL). We used DIP to reconstruct the image using only these representations. As we can see, this method produces most of the time unrealistic images. Figure from Bordes et al. [31]

1.5.1. Interpretability and qualitative evaluations

One of the ways to make neural networks more interpretable is by visualizing what input information they use for classifying a specific object. We can imagine that for classifying birds, the bird's characteristics should be leveraged while the background should be ignored. One way to visualize which pixels are the most important in a given image is to *reverse* the neural network such that instead of predicting the most probable class label given an input, we are predicting the most probable input pixels given a class label. This can be quickly done by computing the input's gradient with respect to a given output activation (which corresponds to a specific class label) and by iteratively applying gradient descent over the input to maximize the output activation. There are many works [73, 237, 188, 150, 183, 190, 160] that rely on such gradient-based techniques to visualize what is learned by neural networks. Some of them maximize only the activation of a specific neuron to visualize what is learned by this neuron. Others offer visualization of what is learned at different layers by trying to "invert" neural networks. All of these use some form of regularization, constraint, or prior to guiding the optimization process towards *realistic* images. Another possibility, explored in Zhao et al. [245], Appalaraju et al. [5], Ericsson et al. [74], is to learn to invert the DN features through a Deep Image Prior (DIP) model (which was one of the state-of-the-art methods for interpretability before our work on RCDM). In short, given a mapping f that produces a representation of interest, the DIP model g_θ learns $\min_\theta d(g_\theta(f(\boldsymbol{x})),\boldsymbol{x})$. However, as demonstrated in Figure 9, this solution not only requires solving a costly optimization problem for each generated sample but also leads to low-quality generation. In Chapter 2, we

introduced a new state-of-the-art interpretability method that significantly outperformed all previous approaches, such as DIP, regarding image quality.

1.5.2. How to evaluate deep neural networks?

Deep neural networks are often built to produce a lower dimensional output from a high dimensional input by going through multiple layers that reduce the input's dimensionality. The deepest representation is often the one to which the training criterion is applied. Consequently, leveraging this deepest representation to solve downstream tasks is the standard way to evaluate neural networks. However, this is not the case in Self-Supervised Learning. Researchers and practitioners systematically discarded the projector representation (the deepest representation) and used only the backbone layer for downstream evaluation (often 3 or 4 layers before the projector representation). Such practice might be surprising since one would expect the deepest representation to learn the correct invariances to solve the task. However, such a trick was justified because the invariances learned with SSL might be too strong [238]. But in the following work, Chen et al. [50] demonstrated that intermediate projector layers might be more suited in a semi-supervised setting. So even if the invariances might *too strong*, there is some setting in which throwing away the entire projector might be a bad idea.

When looking outside SSL, several papers in the transfer learning literature [161, 149] already emphasized that using intermediate representations might be more suited depending on a given downstream task. However, such results are often restricted to a given setting and do not explain why the optimal layer for a downstream task might change. In addition, some SSL methods have used a two-layer projector with 2048 artificial neurons while others have used a three-layer projector with 8192 artificial neurons. These differences in design made the comparison between different SSL criteria unfair since the number of layers used in the projector significantly impacts the generalization abilities of an SSL model. In Chapter 3, we argue that it is important to carefully evaluate the performances of different SSL models across multiple layers of representation to find the optimal one. In addition, we provide significant insights about why and when the optimal layer to use for a given downstream task might change.

1.5.3. Are deep neural networks memorizing?

Memorization in deep neural networks is often associated with over-fitting. A neural network can memorize to associate a specific image of a cat in a given background with the particular cat class without correctly classifying a cat in a different background. This lack of generalization can be perceived as the network being able to solve the task without learning

the features needed to understand what a cat is. In that instance, the neural network perfectly solves the training task on the training data while performing poorly on the validation or test set. However, not learning useful features might not always indicates that a given network is indeed *memorizing* its training data. It could have learned to solve the task by relying on spurious features such that the network might generalize well on images that contain these spurious features but will not generalize on images that do not have them. In that case, it is unclear how much the network memorizes in contrast to learning incorrect features or spurious correlations. Without precise annotations, it might be challenging to disentangle the memorization of training points from learning spurious correlations.

In this thesis, I will try to disentangle learning correlations (even spurious) from memorization by stating that a neural network is memorizing the training set only if it is possible, using this neural network, to reconstruct specific information that is unique to the training images (meaning that they cannot be predicted through simple correlations). There have been many works in the NLP literature around reconstructing training data [37, 18, 94] with some extension to images and diffusion models[38]. In our work, we are the first to show that Self-Supervised Models can memorize their training data beyond any correlations. In Chapter 4, we introduce the DéjaVu score, a memorization metric that can evaluate how much invariance-based methods are memorizing their training data.

1.5.4. Current datasets and benchmarks limitations

A popular benchmark to evaluate neural networks is ImageNet[57] which consists of around 1M images scraped from the internet. Each image is associated with a class label representing one of the 1000 categories. This classification benchmark has been used for the last ten years and has become almost a requirement when publishing a paper on deep learning. However, there are several shortcomings in using a benchmark like ImageNet. The images contain people who did not consent to appear in this dataset and might also contain copyrighted content. The label information is also limited to a class label which is an issue for assessing what a neural network is learning. A network trained on ImageNet can rely on the sky background to classify planes or birds. In that instance, one will require additional annotations to assess the performance of a given model fully. But getting more annotations on real data is a complex process that relies on annotators. However, there is difficulty in having consistency and how fine-grained we want the annotations to be. Also, some annotators might be incentivized to complete the annotation task as fast as possible without considering the annotations' quality.

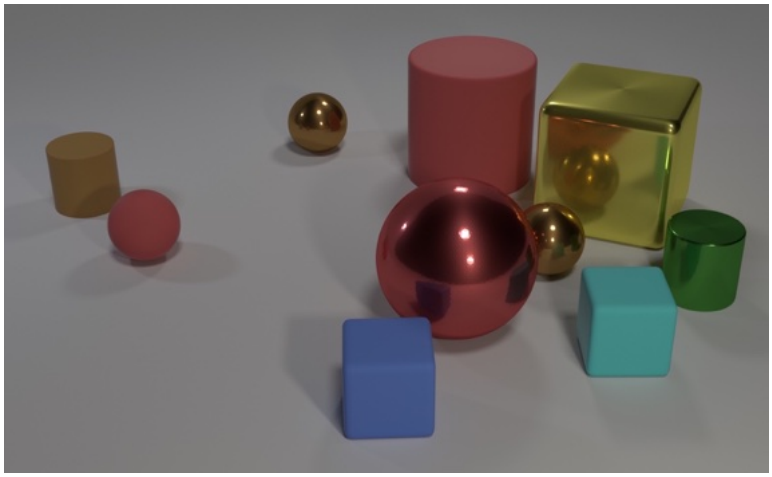


Fig. 10. Exemple of synthetic data from the CLEVR dataset. Figure from Johnson et al. [118]

In contrast, synthetic data offer an excellent platform to produce data with reliable annotations. Compared to neural networks that are *black box*, a rendering engine can be seen as a *white box* in which we can specify everything inside the scene. However, the usage of synthetic data is not as popular as benchmarks relying on real data such as ImageNet. Synthetic datasets like CLEVR[118] are used for question-answering and visual reasoning. However, as shown in Figure 10, the scope of the dataset is limited to elementary geometric forms. Other approaches try to leverage synthetic data but often rely on simple rendering engines that do not produce realistic images. The gap between synthetic images generated with rendering engines and real images was wide enough that there was an entire part of the research literature around bridging the *sim-to-real* gap. However, in recent years, rendering engines have made such significant progress that it is possible today to render digital and completely photorealistic clones of a real environment. In Chapter 5, we introduce our new PUG framework and photorealistic datasets for which neural networks get good zero-shot performances without needing additional tricks to bridge a sim-to-real gap.

Prologue to Article 1

High Fidelity Visualization of What Your Self-Supervised Representation Knows About, *Florian Bordes, Randall Balestriero, and Pascal Vincent*, Transactions on Machine Learning Research, July 2022.

The original motivation for this work was to develop a superior tool to visualize what information might be contained in Self-Supervised representations. This was something that Pascal wanted to have as he was interested in exploring augmentations in representation space for self-supervised learning. Before I joined, Pascal and others tried to learn deterministic decoders from the representation space to visualize reconstructions. However, the main challenge they faced was that these decoders were only able to produce blurry reconstructions. One hypothesis was that there is a need to handle uncertainty when generating images from an SSL representation. Hence, I started my experiments with diffusion models because they learn to model, and stochastically produce, a distribution of images. They were known to be stable to train yet capable of producing sharp images. In order to find a way to condition the diffusion model on an SSL representation, I drew inspiration from my previous work on Infusion Generative Modeling[28] in which I used conditional batch normalization to add extra conditioning when training the model. Once this conditioning was implemented, training the model was straightforward.

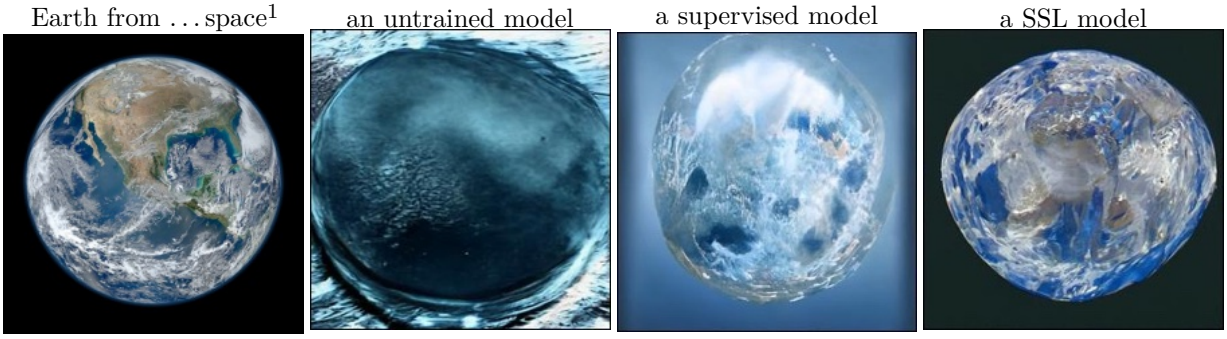
Contribution statement The original idea for this work arose during discussions between Pascal and I. I developed the core method, which is a diffusion model conditioned on a rich SSL latent representation – while about all preexisting diffusion model work conditioned on a class label or on class-based information. I wrote all the code – that has been made public in the GitHub repository for the project. I ran all the experiments which are presented in the paper. I created all the figures, wrote the captions and most of the paper's content. Pascal and Randall's feedback was very helpful in improving the writing of the paper. I also handled and wrote most of the answers to reviewers during the review process for TMLR.

Impact of this work Even if this work was mostly designed to help with the interpretability of SSL methods, it also had a significant impact on the generative model research community. DALL · E 2 [167] from OpenAI, which drew a lot of attention from the public, was inspired by our work. DALL-E 2 and RCDM are almost the same models except that instead of using a pre-trained SSL models trained on images, OpenAI has used a pre-trained SSL model trained on text as conditioning (which gives them the ability to control the generation through text prompts). Concerning Self-Supervised Learning, RCDM was an important component behind the conception of MSN [9] since it allowed us to understand that the prototypes learned by this model were class-specific. RCDM has also provided important visualization for the following works: Assran et al. [8], Bordes et al. [30], Assran et al. [11], Shekhar et al. [184]. RCDM was also used as a tool in subsequent Articles 2 and 3.

Chapter 2

Article 1: High Fidelity Visualization of What Your Self-Supervised Representation Knows About

Discovering what is learned by neural networks remains a challenge. In self-supervised learning, classification starting with the learned representation as input is the most common task used to evaluate how good a representation is. However, relying only on such downstream task can limit our understanding of what information is retained in the representation of a given input. In this work, we showcase the use of a Representation Conditional Diffusion Model (RCDM) to visualize in data space the representations learned by self-supervised models. The use of RCDM is motivated by its ability to generate high-quality samples —on par with state-of-the-art generative models— while ensuring that the representations of those samples are faithful i.e. close to the one used for conditioning. By using RCDM to analyze self-supervised models, we are able to clearly show visually that i) SSL (backbone) representation are *not* invariant to the data augmentations they were trained with – thus debunking an often restated but mistaken belief; ii) SSL post-projector embeddings appear indeed invariant to these data augmentation, along with many other data symmetries; iii) SSL representations appear *more robust to small adversarial perturbation* of their inputs than representations trained in a supervised manner; and iv) that SSL-trained representations exhibit an inherent structure that can be explored thanks to RCDM visualization and enables image manipulation. Code and trained models are available at `https://github.com/facebookresearch/RCDM`.



2.1. Introduction

Approaches aimed at learning useful representations, from unlabeled data, have a long tradition in machine learning. These include probabilistic latent variable models and variants of auto-encoders [2, 104, 177, 213, 121, 168], that are traditionally put under the broad umbrella term of *unsupervised learning* [27]. More recent approaches, under the term of *self-supervised learning* (SSL) have used various kinds of "pretext-tasks" to guide the learning of a useful representations. Filling-in-the-blanks tasks, proposed earlier in [213, 215], later proved remarkably successful in learning potent representations for natural language processing [211, 58]. Pretext tasks for the image domain include solving Jigsaw-puzzles [158], predicting rotations or affine transformations [84, 241] or discriminating instances [224, 209]. The latest, most successful, modern family of SSL approaches for images [154, 49, 51, 97, 93, 40, 44, 236, 21], have two noteworthy characteristics that markedly distinguish them from traditional unsupervised-learning models such as autoencoder variants or GANs [88]: a) their training criteria are not based on any input-space reconstruction or generation, but instead depend only on the obtained distribution in the representation or embedding space b) they encourage invariance to explicitly provided input transformations a.k.a. data-augmentations, thus injecting important additional domain knowledge.

Despite their remarkable success in learning representations that perform well on downstream classification tasks, rivaling with supervised-trained models [49], much remains to be understood about SSL algorithms and the representations they learn. How do the particularities of different algorithms affect the representation learned and its usefulness? What information does the learned representation contain? Answering this question is the main focus of our work. Since SSL methods mostly rely on learning invariances to a specific set of handcrafted data augmentations, being able to evaluate these invariances

[1]We use representations of the real picture of Earth on the left (source: NASA) as conditioning for RCDM. We show samples (resolution 256×256) in cases where the representations (2048-dimensions) were obtained respectively with a random initialized ResNet50, a supervised-trained one, and a SSL-trained one. More samples in Fig. 31.

will provide insight into how successful the training of the SSL criteria has been. It is also worth to be noted that recent SSL methods use a **projector** (usually a small MLP) on top of a **backbone** network (resnet50 or vit) during training, where the projector is usually discarded when using the model on downstream tasks. This projector trick is essential to get competitive performance on ImageNet i.e we often observe a performance boost of 10 to 30 percentage accuracy point when using the representation at the backbone level instead of the ones at the projector level on which SSL criteria are applied during training. Since SSL criteria are not applied at the backbone level, there remains a mystery regarding what the backbone does learn that make it better for classification than the projector. This is another question that we will be able to answer in this study.

Empirical analyses have so far attempted to analyse SSL algorithms almost exclusively through the limited lens of the numerical performance they achieve on downstream tasks such as classification. *Contrary to their older unsupervised learning cousins, modern SSL methods do not provide any direct way of mapping back the representation in image space, to allow* visualizing *it. The main goal of our work is thus to enable the visualization of representations learned by SSL methods, as a tool to improve our understanding.*

In our approach (Section 2.3), we propose to generate samples conditioned on a representation such that (i) the representation of those samples match the one used for conditioning, and (ii) the visual quality of the sample is as high as possible to maximize the preciseness of our visual understanding. For this, we employ a conditional generative model that (implicitly) models. For reasons that we will explain later, we opted for a conditional diffusion model, inspired by Dhariwal and Nichol [60].

This paper's main contributions are:

- To demonstrate how recent diffusion models are suitable for conditioning on large vector representations such as SSL representations. The conditionally generated images, in addition to being high-quality are also highly representation-faithful i.e. they get encoded into a representation that closely matches the representation of the images used for the conditioning (Tab. 1b, Fig. 18).
- To showcase its potential usefulness for qualitatively analyzing SSL representations and embeddings (also in contrast with supervised representations), by shedding light on what information about the input image is or isn't retained in them.

Specifically, by repeatedly sampling from a same conditioning representation, one can observe which aspects are common to all samples, thus identifying what is encoded in the representation, while the aspects that vary greatly show what was *not retained* in the representation. We make the following observations: (i) SSL projector embeddings appear

most invariant, followed by supervised-trained representation and last SSL representations[2] (Fig. 2). (ii) SSL-trained representations retain more detailed information on the content of the background and object style while supervised-trained representations appear oblivious to these (Fig. 3). (iii) despite the invariant training criteria, SSL representations appear to retain information on object scale, grayscale vs color, and color palette of the background, much like supervised representation (Fig. 3). (iv) Supervised representations appear more susceptible to adversarial attacks than SSL ones (Fig. 4,24). (v) We can explore and exploit structure inside SSL representations leading to meaningful manipulation of image content (such as splitting representation in foreground/background components to allow background substitution) (Fig, 5, 25, 26).

2.2. Related Work

Visualization methods: Many works [73, 237, 188, 150, 183, 190, 160] used gradient-based techniques to visualize what is learned by neural networks. Some of them maximize the activation of a specific neuron to visualize what is learned by this neuron, others offer visualization of what is learned at different layers by trying to "invert" neural networks. All of these use some form of regularization, constraint or prior to guide the optimization process towards *realistic* images. Dosovitskiy and Brox [63] learn to map back a representation to the input space by using a Generative Adversarial Networks [88] which is trained to reconstruct an input given a representation. Since the mapping is deterministic, they obtain only a single image with respect to a specific conditioning. In contrast, we use a stochastic mapping that allows us to visualize the diversity of the images associated to a specific representation. Nguyen et al. [156] also use GANs but instead of trying to invert the entire vector of representation, they try to find which images (by using an optimization process in the latent space of the generator) maximize a specific neuron. The following work [157] demonstrates how using this conditional iterative optimization in the latent space of the generator lead to high quality conditional image generation. Finally Lučić et al. [146] leverage SSL methods to create discrete cluster that are used as conditioning for a GAN. Our work focuses only on continuous representation vectors. More recently, Caron et al. [44] used the attention mask of transformers to perform unsupervised object segmentation. By contrast, our method is not model dependent, we can plug any type of representation as conditioning for the diffusion model. Another possibility, explored in Zhao et al. [245], Appalaraju et al. [5], Ericsson et al. [74], is to learn to invert the DN features through a Deep Image Prior (DIP) model. In short, given a mapping f that produces a representation of interest, the DIP model g_θ learns $\min_\theta d(g_\theta(f(\boldsymbol{x})),\boldsymbol{x})$. However, as we will demonstrate in Figure 2 this solution not only requires to solve an optimization problem for each generated sample but also leads to

[2]The representation that is produced by a Resnet50 backbone, before the projector.

low-quality generation.

Generative models: Several families of techniques have been developed as generative models, that can be trained on unlabeled data and then employed to generate images. These include auto-regressive models [208], variational auto-encoders [121, 168], GANs [88], autoregressive flow models [122], and diffusion models [191]. Conditional versions are typically developed shortly after their unconditional versions [152, 207]. In principle one could envision training a conditional model with any of these techniques, to condition on an SSL or other representation for visualization purpose, as we are doing in this paper with a diffusion model. One fundamental challenge when conditioning on a rich representation such as the one produced by a SSL model, is that for a given conditioning $\boldsymbol{h}$ we will usually have available only a *single* corresponding input instance $\boldsymbol{x}$, By contrast a particularly successful model such as the conditional version of BigGAN [35] conditions on a categorical variable, the class label, that for each value of the conditioning has a large number of associated $\boldsymbol{x}$ data. One closely related work to ours is the recent work on Instance-Conditioned GANs (IC-GAN) of Casanova et al. [46]. Similar to us it also uses SSL or supervised representations as conditioning when training a conditional generative model, here a GAN [88], specifically a variant of BigGAN [35] or StyleGAN2 [119]. However, the model is trained such that, from a specific representation, it learns to generate not only images that should map to this representation, but a much broader neighborhood of the training data. Specifically up to 50 training points that are its nearest neighbors in representation space. It remains to be seen whether such a GAN architecture could be trained successfully without resorting to a nearest neighbor set. IC-GAN is to be understood as a conditional generative model of an image's broad *neighborhood*, and the primary focus of the work was on developing a superior quality controllable generative model. By contrast we want to sample images that map as closely as possible to the original image in the representation space, as our focus is to build a tool to analyse SSL representations, to enable visualising what images correspond *precisely* to a representation. (See Fig. 18 for a comparison.)

Few approaches have focused on conditional generation to unravel the information encoded in representations of supervised models. In Shocher et al. [185], a hierarchical LSGAN generator is trained with a class-conditional discriminator [240]. While the main applications focused on inpainting and style-transfer, this allowed to visually quantify the increasing invariance of representations associated to deeper and deeper layers. This method however requires labels to train the generator. On the other hand, Nash et al. [155] proposed to use an autoregressive model, in particular PixelCNN++ [180], to specifically study the invariances that each layer of a DN inherits. In that case, the conditioning was incorporated by regressing a context vector to the generator biases. As far as we are aware, PixelCNN++ generator

falls short on high-resolution images e.g. most papers focus on 32×32 Imagenet. Lastly, Rombach et al. [169] proposes to learn a Variational Auto Encoder (VAE) that is combined with an invertible neural network (INN) whose role is to model the relation between the VAE latent space and the given representations. To allow for interpretable manipulation, a second invertible network [75] is trained using labels to disentangle the factors of variations present in the representation. By contrast we train end-to-end a single decoder to model the entire diversity of inputs that correspond to the conditioning representation, without imposing constraints of a structured prior or requiring labels for image manipulation.

2.3. High-Fidelity Conditioning with Diffusion Models

We base our work on the Ablated Diffusion Model (ADM) developed by Dhariwal and Nichol [60] which uses a UNet architecture [170] to learn the reverse diffusion process. Our conditional variant – called *Representation-Conditionned Diffusion Model* (RCDM) – is illustrated in Fig. 1b. To suitably condition on representation $\boldsymbol{h} = f(\boldsymbol{x})$, we followed the technique used by Casanova et al. [46] for IC-GAN which rely on conditional batch normalization [68]. More precisely, we replaced the Group Normalization layers of ADM by conditional batch normalization layers that take $\boldsymbol{h}$ as conditioning. We also apply a fully connected layer to $\boldsymbol{h}$ that reduces dimension to a vector of size 512. This vector is then given as input to multiple conditional batch normalization layers that are placed in each residual block of the diffusion model. An alternative conditioning method is to use the original conditioning method built inside ADM, and replace the embedding vector used for class labels by a linear layer that reduces dimension to a vector of the size of the time steps embedding. We didn't observe any differences in term of experimental results between those two methods as presented in Fig. 5. In this paper, we used the first conditioning method for most of the experiments in order to make a proper comparison with IC-GAN.

In contrast with Dhariwal and Nichol [60] we don't use the input gradient of a classifier to bias the reversed diffusion process towards more probable images (classifier-guidance), nor do we use any label information for training our model – recall that our goal is building a visualization tool for SSL models that train on unlabeled data. Another drawback of classifier-guidance is the need to retrain a classifier on inputs generated by the diffusion process. Since training SSL models can be very costly, retraining them was not an option, thus we had to find a method that could use directly the representation of a pretrained model.

Our first experiments aim at evaluating the abilities of our model to generate realistic-looking images whose representations are close to the conditioning. To do so, we trained our

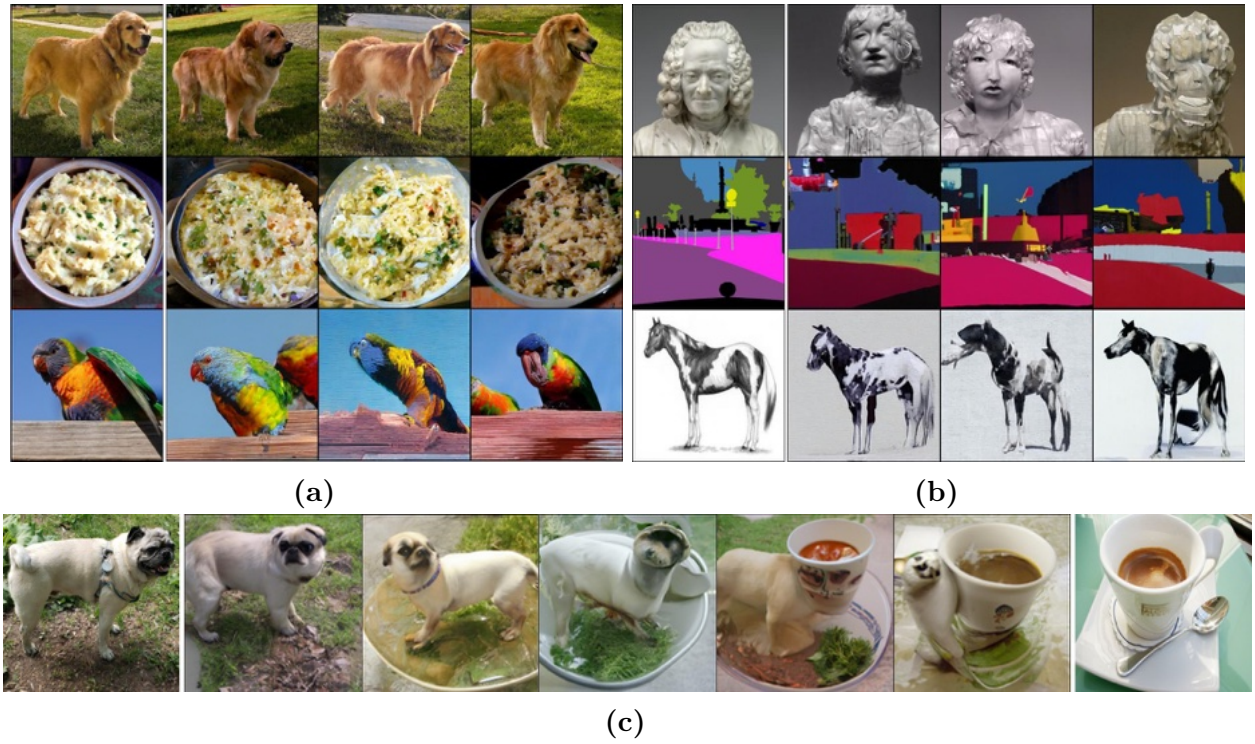


Fig. 1. a) In-distribution conditional image generation. An image from ImageNet validation set (first column) is used to compute the representation output by a trained SSL model (Dino backbone). The representation is used as conditioning for the diffusion model. Resulting samples are shown in the subsequent columns (see Fig. 3). We observe that our conditional diffusion model produces samples that are very close to the original image. b) Out of distribution (OOD) conditioning. How well does RCDM generalize when conditioned on representations given by images from a different distribution? (here a WikiMedia Commons image, see Fig. 4 for more). Even with an OOD conditioning, the images produced by RCDM match some characteristics of the original image (which highlights that RCDM is not merely overfitting on ImageNet). c) Interpolation between two images from ImageNet validation data. We apply a linear interpolation between the SSL representation of the images in the first column and the representation of the images in the last column. We use the interpolated vector as conditioning for our model, that produces the samples that are showed in columns 2 to 6. Fig. 10 in appendix shows more sampled interpolation paths.

Representation-Conditionned Diffusion Model (RCDM), conditioned on the 2048 dimensional representation given by a Resnet50 [95] trained with Dino [44] on ImageNet [174]. Then we compute the representations of a set of images from ImageNet validation data to condition the sampling from the trained RCDM. Fig. 1a shows it is able to sample images that are very close visually from the one that is used to get the conditioning. We also evaluated the generation abilities of our model on out of distribution data. Fig. 1b shows that our model is able to sample new views of an OOD image. We also quantitatively validate that the generated images' representations are close to the original image representation in Tab. 1b, Fig. 12, Fig. 13 and Fig. 14. We do so by verifying that the representation used as conditioning is the nearest neighbor of the representation of its generated sample.

(a) We report results for ImageNet to show that our approach is reliable for generating images which look realistic. Since the focus of our work is not generative modelling but to showcase and encourage the use of such model for representation analysis, we only show results for one conditional generative models. For each method, we computed FID and IS with the same evaluation setup in Pytorch.

Method	Res.	↓FID	↑IS
ADM [60]	256	26.8	34.5 ± 1.8
IC-GAN [46])	256	20.8	51.3 ± 2.2
IC-GAN [46] w/ KDE*	256	21.6	38.6 ± 1.1
RCDM w/ KDE* (ours)	256	19.0	51.9 ± 2.6

(b) For each encoder, we compute the rank and mean reciprocal rank (MRR) of the image used as conditioning within the closest set of neighbor in the representation space of the samples generated from the valid set (50K samples). A rank of one means that all of the generated samples for a given model have their representations matching the representation used as conditioning.

Model	↓Mean rank	↑MRR
Dino [44]	1.00	0.99
Swav [40]	1.01	0.99
SimCLR [49]	1.16	0.97
Barlow T. [236])	1.00	0.99
Supervised	5.65	0.69

Table 1. a) Table of results on ImageNet. We compute the FID [103] and IS [179] on 10 000 samples generated by each models with 10 000 images from the validation set of ImageNet as reference. KDE* means that we used the unconditional representation sampling scheme based on KDE (Kernel Density Estimation) for conditioning IC-GAN instead of the method based on K-means introduces by Casanova et al. [46]. b) Table of ranks and mean reciprocal ranks for different encoders. This table show that RCDM is faithful to the conditioning by generating images which have their representations close to the original one.

This implies that there is much information kept inside the SSL representation so that the conditional generative model is able to reconstruct many characteristics of the original image. We also perform interpolations between two SSL representations in Fig. 1c. This shows that our model is able to produce interpretable images even for SSL representations that correspond to an unlikely mix of factors. Both the interpolation and OOD generation clearly show that the RCDM model is not merely outputting training set images that it could have memorized. This is also confirmed by Fig. 11 in the appendix that shows nearest neighbors of generated points.

The conditional diffusion model might also serve as a building block to hierarchically build an unconditional generative model. Any technique suitable for modeling and sampling the distribution of (lower dimensional) representations could be used. As this is not our primary goal in the present study, we experimented only with simple kernel density estimation (see appendix A.2). This allow us to quantify the quality of our generative process in an unconditional manner to fairly compare against state-of-the-art generative models such as ADM; we provide some generative model metrics in Tab. 1a along some samples in Fig. 3 to show that our method is competitive with the current literature, even in unconditional generation setting.

Model	SimCLR Trunk	SimCLR Head	Dino Trunk	Dino Head	Barlow T. Trunk	Barlow T. Head	VicReg Trunk	VicReg Head
Val acc.	69.1 %	61.2 %	74.8 %	64.9 %	72.6 %	62.9 %	72.3 %	62.2 %

Table a): ImageNet linear probe validation accuracy on representation given by various SSL models. We observe an accuracy gap between the linear probes at the trunk level and the linear probes trained at the head level of around 10 percentage point of accuracy.

Fig. 2. What is encoded inside various representations? First to fourth rows show RCDM samples conditioned on the usual resnet50 backbone representation (size 2048) while fifth to eigth rows show samples conditionned on the projector/head representation of various ssl models. (Note that a separate RCDM generative model was trained specifically for each representation). *Common/stable aspects* among a set of generated images reveal *what is encoded* in the conditioning representation. *Aspects that vary* show *what is not encoded* in the representation. We clearly see that the projector representation only keeps global information and not its context, contrary to the backbone representation. This indicates that invariances in SSL models are mostly achieved in the projector representation, not the backbone. Furthermore, it also confirms the linear classification results of Table a) which show that backbone representation are better for classifications since they contain more information about an input than the ones at the projector level. Additional comparisons provided in Fig. 19.

2.4. Visual Analysis of Representations Learned by Self-Supervised Model

The ability to view generated samples whose representations are very close in the representation space to that of a conditioning image can provide insights into what's hidden in such a representation, learned by self-supervised models. As demonstrated in the previous section, the samples that are generated with RCDM are really close visually to the image used as conditioning. This gives an important indication of how much is kept inside a SSL representation in general. However, it is also interesting to see how much this amount of "hidden" information varies depending on what specific SSL representation is being considered. To this end we train several RCDM on SSL representations given by VicReg [21], Dino [44], Barlow Twins [236] and SimCLR [49]. In many applications that use self-supervised models, the representation that is subsequently used is the one obtained at the level of the backbone of the ResNet50. Usually, the representation computed by the projector of the SSL-model (on which the SSL criterion is applied) is discarded because the results on many downstream tasks like classification is not as good as when using the backbone representation. However, since our goal is to visualize and better understand the differences between various SSL representations, we also trained RCDM on the representation given by the projector.

In Fig. 2 we condition all the RCDM with the image labelled as conditioning and sample 7 images for each model. We observe that the representation at the backbone level does not allow much variance in the generated samples. Even information about the pose and size of the animal is kept in the representation. In contrast, when looking at the samples generated by using representations at the projector level (also coined as head in the figure), we observe much more variance in the generated samples, which indicates that significant information about the input has been lost. These qualitative differences are correlated [3] with the quantitative experiment we made in Fig. 2 Table a) highlighting that when training a linear probe over the corresponding representations, the performances at the backbone level are better than the ones at the projector level.

2.4.1. Are Self-Supervised Representations Really Invariant to Data-Augmentations?

In Fig. 3, we apply specific transformations (augmentations) to a test image and we check whether the samples generated by the diffusion model change accordingly. We also compare with the behavior of a supervised model. We note that despite their invariant training criteria, the 2048 dimensional SSL representations do retain information on object scale,

[3]This point is further demonstrated in Figure 7 in the Appendix.

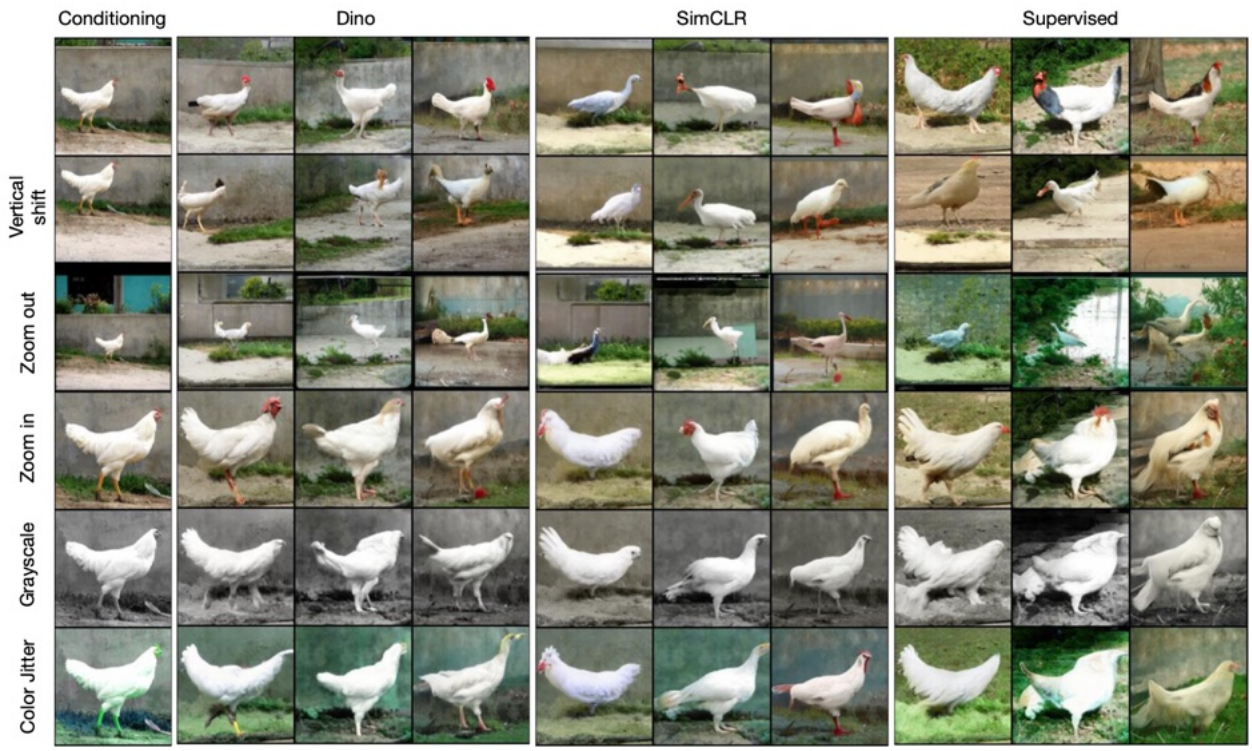


Fig. 3. Using our conditional generative model to gain insight about the invariance (or covariance) of representations with respect to several data augmentations. On an original image (top left) we apply specific transformations (visible in the first column). For each transformed image, we compute the 2048-dimensional representation of a ResNet50 backbone trained with either Dino, SimCLR, or a fully supervised training. We then condition their corresponding RCDM on that representation to sample 3 images. We see that despite their invariant training criteria, the 2048 dimensional SSL representations appear to retain information on object scale, grayscale vs color, and color palette of the background, much like the supervised-trained representation. They do appear insensitive to vertical shifts. We also see that supervised representation constrain the appearance much less. Refer to Fig. 20 in Appendix for a comparison with using the lower dimensional projector-head embedding as the conditioning representation.

grayscale status, and color palette of the background, much like the supervised representation. They do appear invariant to vertical shifts. In the Appendix, Fig. 20 applies the same transformations, but additionally compares using the 2048 representation with using the lower dimensional projector head embedding as the representation. There, we observe that the projector representation seems to encode object scale, but contrary to the 2048 representation, it appears to have gotten rid of grayscale-status and background color information. Currently, researchers need to use custom datasets (in which the factors of variation of a specific image are annotated) to verify how well the representations learned are invariant to those factors. We hope that RCDM will help researchers in self-supervised learning to alleviate this concern since our method is "plug and play" and can be easily used on any dataset with any type of representation.

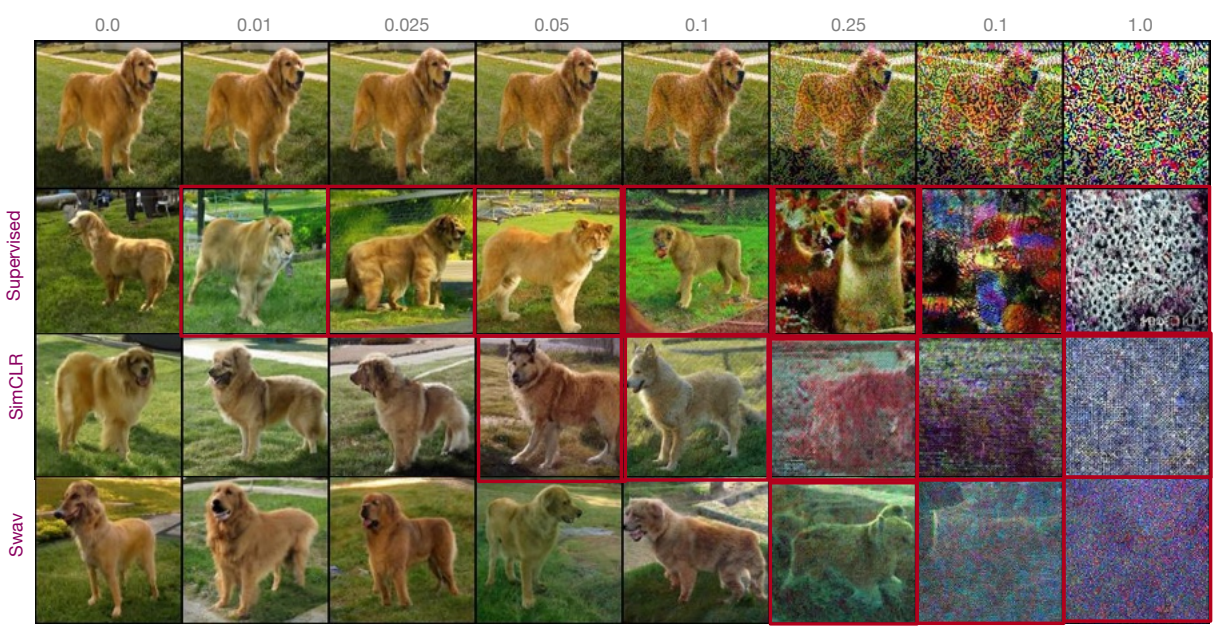


Fig. 4. Using RCDM to visualize the robustness of differently-trained representations to adversarial attacks. We use Fast Gradient Sign to attack a given image (top-left corner) on different models with various values for the attack coefficient epsilon. In the first row, we only show the adversarial images obtained from a supervised encoder: refer to Fig. 24 in the Appendix to see the (similar looking) adversarial examples obtained for each model. In the following rows we show, for differently trained models, the RCDM "stochastic reconstructions" of the adversarially attacked images, from their ResNet-50 backbone representation. For an adversarial attack on a purely supervised model (second row), RCDM reconstructs an animal that belongs to another class, a lion in this case. Third and forth rows show what we obtain with ResNet50 that was pretrained with SimCLR or Swav in SSL fasion, with only their linear softmax output layer trained in a supervised manner. In contrast to the supervised model, with the SSL-trained models, RCDM stably reconstructs dogs from the representation of adversarially attacked inputs, even with quite larger values for epsilon. Images classified incorrectly by a trained linear probe are highlight with a red square.

2.4.2. Self-Supervised and Supervised Models See Adversarial Examples Differently

Since our model is able to "project back" representations to the manifold of realistic-looking images, we follow the same experimental protocol as Rombach et al. [169] to visualize how adversarial examples affect the content of the representations, as seen through RCDM. We apply Fast Gradient Sign attacks (FGSM) [89] over a given image and compute the representation associated to the attacked image. When using RCDM conditioned on the representation of the adversarial examples, we can visualize if the generated images still belong to the class of the attacked image or not. In Fig. 4 and 24, the adversarial attacks change the dog in the samples to a lion in the supervised setting whereas SSL methods doesn't seem to be impacted by the adversarial perturbations i.e the samples are still dogs until the adversarial attack became visible to the human eye.

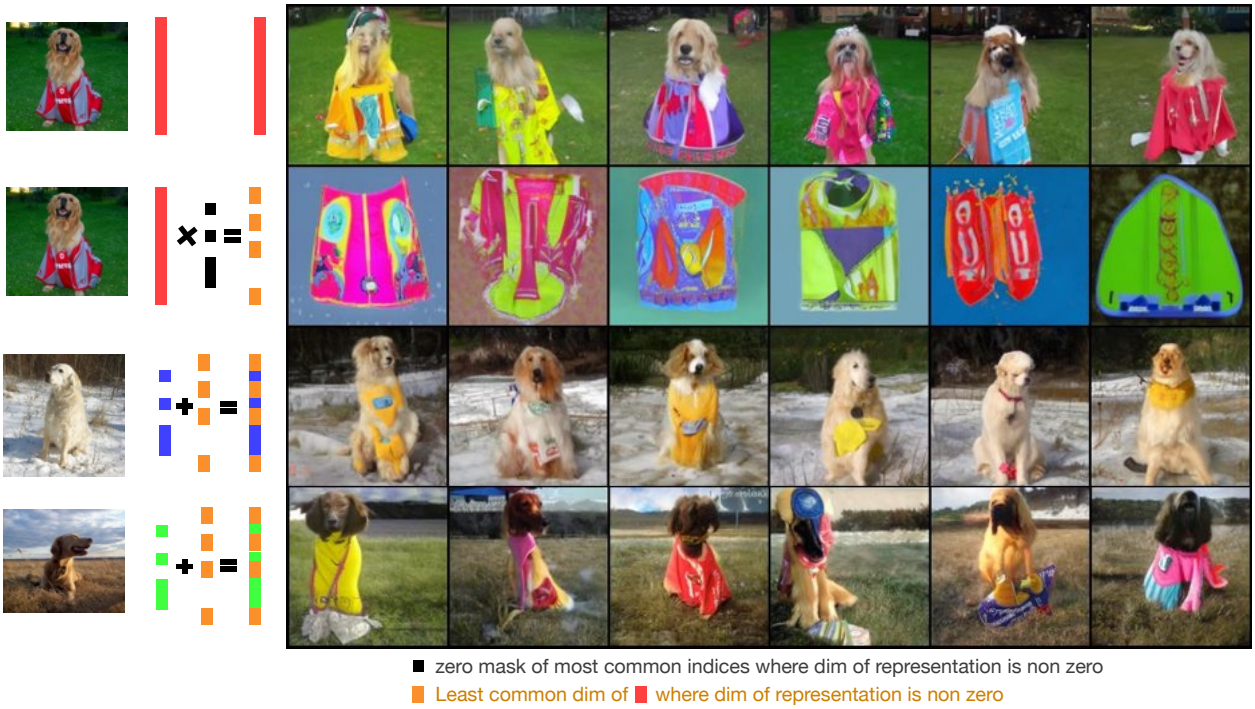


Fig. 5. Visualization of direct manipulations in the representation space of a ResNet-50 backbone trained with SimCLR. In this experiment, we find the most common non-zero dimensions among the neighborhood (in representation space) of the image used as conditioning (top-left clothed dog). In the second row, we set these dimensions to zero and use RCDM to decode the thus masked representation. We see that RCDM produces a variety of clothes (but no dog): all information about the background and the dog has been removed. In the third and forth row, instead of setting these dimensions to zero, we set them to the value they have in the representation of the unclothed-dog image on the left. As we can see, the generated dog gets various clothes which were not present in the original image. Additional examples provided in Figure 25, 26.

2.4.3. Self-Supervised Representations Locally Encode background and Object on Different Dimensions

Experimental manipulation of representations can be useful to analyze to what degree specific dimensions of the representation can be associated with specific aspects or factors of variations of the data. In a self-supervised setting in which we don't have access to labelled data, it can be difficult to gain insight as to how the information about the data is encoded in the representation. We showcase a very simple and heuristic setup to remove the most common information in the representations within a set of the nearest neighbors of a specific example. We experimentally saw that the nearest neighbors of a given representation share often similar factors of variation. Having this information in mind, we investigate how many dimensions are shared in between this set of neighbors. Then, we mask the most common non-zero dimensions by setting them to zero and use RCDM to decode this masked representation. In Fig. 5, this simple manipulation visibly yields the removal of all information about the background and the dog, to only keep information about clothing (only one dog had clothes in the set of neighbors used to find the most common dimensions).

Since the information about the dog and the background are removed, RCDM produces images of different clothes only. In the third and fourth row, instead of setting the most common dimensions to zeros, we set them to the value they have in other unclothed dog images. By using these new representations, RCDM is able to generate the corresponding dog with clothes. This setup works better with SSL methods, as supervised models learn to remove from their representation most of the information that is not needed to predict class labels. We show a similar experiment for background removal and manipulation in Figure 25 in the Appendix.

2.5. Conclusion

Most of the Self-Supervised Learning literature uses downstream tasks that require labeled data to measure how good the learned representation is and to quantify its invariance to specific data-augmentations. However one cannot in this way see the entirety of what is retained in a representation, beyond testing for specific invariances known beforehand, or predicting specific labeled factors, for a limited (and costly to acquire) set of labels. Yet, through conditional generation, all the stable information can be revealed and discerned from visual inspection of the samples. We showcased how to use a simple conditional generative model (RCDM) to visualize representations, enabling the visual analysis of what information is contained in a self-supervised representation, without the need of any labelled data. After verifying that our conditional generative model produces high-quality samples (attested qualitatively and by FID scores) and representation-faithful samples, we turned to exploring representations obtained under different frameworks. Our findings clearly separate supervised from SSL models along a variety of aspects: their respective invariances – or lack thereof – to specific image transformations, the discovery of exploitable structure in the representation's dimensions, and their differing sensitivity to adversarial noise.

2.6. Reproducibility statement

The data and images in this paper were only used for the sole purpose of exchanging reproducible research results with the academic community.

Our results should be easily reproducible as:

- RCDM, is based on the same code as Dhariwal and Nichol [60] (`https://github.com/openai/guided-diffusion`) and uses the same hyper-parameters (See Appendix I of Dhariwal and Nichol [60] for details about the hyper-parameters).
- To obtain our conditional RCDM, one just needs to replace the GroupNormalization layers in that architecture by a conditional batch normalization layer of Brock et al. [35] (using the code from `https://github.com/ajbrock/BigGAN-PyTorch`).

- The self-supervised pretrained models we used to extract the conditioning representations were obtained from the model-zoo of VISSL [92] (code from `https://github.com/facebookresearch/vissl`).
- The unconditonal sampling process is straightforward, as explained in Appendix A.2.
- We are working on cleaning and preparing to release any remaining code glue to easily reproduce the results in this paper.

2.7. Broader impact statement

Our work aims to promote the use of conditional generative models to project back in image space the internal representation learned by latest and future techniques to train deep artificial neural networks – in order to better understand their inner workings. Such improved understanding through qualitative visualizations, in complement with quantitative metrics, is expected to foster the development of more robust and reliable neural network algorithms. Controlled generation of realistic images is not the goal and focus of this work, as we merely use it as a tool for scientific understanding. Yet conditional generative models have already been and will likely continue to be used and improved to generate fake images, including of synthesized imaginary situations and people, that we expect will be increasingly realistic and hard to impossible to distinguish from real photographs. Such technology will be usable for positive creative pursuits, as well as for voluntarily misleading portrayals of fakes passed as truths and facts.

Prologue to Article 2

Guillotine Regularization: Why removing layers is needed to improve generalization in Self-Supervised Learning, *Florian Bordes, Randall Balestriero, Quentin Garrido, Adrien Bardes, Pascal Vincent*, Transactions on Machine Learning Research, May 2023.

This work's motivation was to better understand why a projector is needed in Self-Supervised Learning. The projector is often a small multi-layer perceptron (with 2 or 3 layers) that is added on top of the network during training and that is immediately discarded once training is done. Discarding the projector is extremely common in Self-Supervised Learning, however, it breaks one of the core goals of SSL which is to learn invariant and structured representations (since the invariance criterion is applied to the projector representation). As shown in the first article, representations at the backbone level do not satsify well the desired invariances while the representations taken at the projector level do. Thus, it is a little vexing that the invariances learned with SSL methods do not offer better projector representations that perform well on downstream tasks. In this article, we explore when and how the invariances directly learned by the projector might be useful for downstream tasks. By empirically demonstrating and evaluating the performances of SSL models across layers, we establish some guidelines to better evaluate SSL models.

Contribution statement With the exception of the experiment in Figure 2 (which was done by Quentin Garrido), all the experiments were done by myself as well as most of the writing. The project was supervised by Pascal Vincent. Randall, Quentin, Adrien, and Pascal helped me with the writing. I wrote most of the rebuttal for TMLR.

Chapter 3

Article 2: Guillotine Regularization: Why removing layers is needed to improve generalization in Self-Supervised Learning

One unexpected technique that emerged in recent years consists in training a Deep Network (DN) with a Self-Supervised Learning (SSL) method and using this network on downstream tasks but *with its last few projector layers entirely removed*. This *trick of throwing away the projector* is actually critical for SSL methods to display competitive performances on ImageNet for which more than 30 percentage points can be gained that way. This is a little vexing, as one would hope that the network layer at which invariance is explicitly enforced by the SSL criterion during training (the last projector layer) should be the one to use for best generalization performance downstream. But it seems not to be, and this study sheds some light on why. This trick, which we name Guillotine Regularization (GR), is in fact a generically applicable method that has been used to improve generalization performance in transfer learning scenarios. In this work, we identify the underlying reasons behind its success and show that the optimal layer to use might change significantly depending on the training setup, the data or the downstream task. Lastly, we give some insights on how to reduce the need for a projector in SSL by aligning the pretext SSL task and the downstream task.

3.1. Introduction

Many recent self-supervised learning (SSL) methods consist in learning invariances to specific chosen relations between samples – implemented through data-augmentations – while using a regularization strategy to avoid collapse of the representations [49, 52, 93, 134, 40, 236, 22, 199, 43, 53, 136, 247, 248]. Incidentally SSL learning frameworks also heavily rely on a simple trick to improve downstream task performances: *removing the last few layers of the trained deep network* depicted in Figure 1a. From a practical viewpoint, this technique emerged naturally [49] through the search of ever increasing SSL performances.

In fact, on ImageNet [56], such technique can improve classification performances by around 30 points of percentage (Figure 1b).

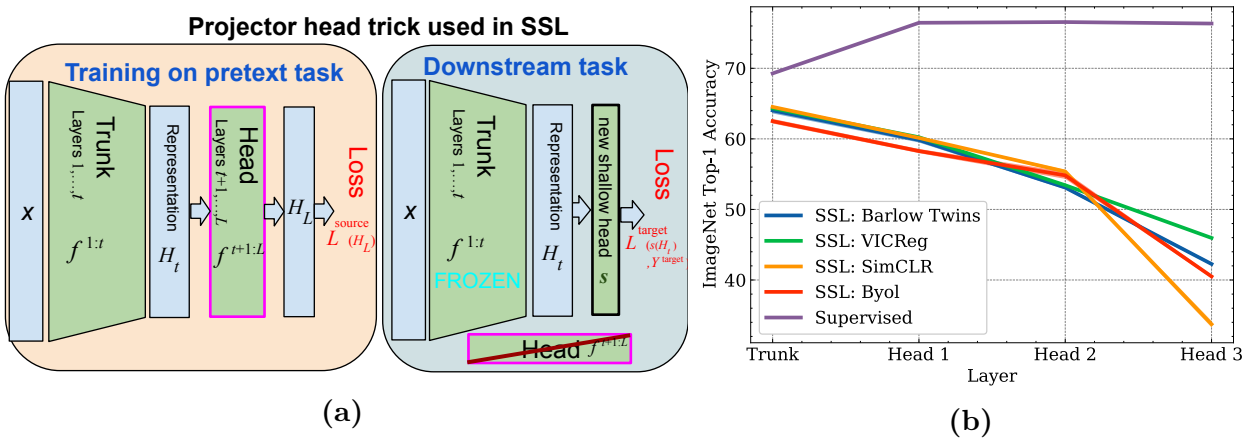
Although it improves performances in practice, not using the layer on which the SSL training was applied is unfortunate. It means throwing away the representation that was explicitly trained to be invariant to the chosen set of data augmentations, thus breaking the implied promise of using a more structured, controlled, invariant representation. By picking instead a representation that was produced an arbitrary number of layers above, SSL practitioners end up relying on a representation that likely contains much more information about the input [29] than should be necessary to robustly solve downstream tasks.

Although the use of this technique emerged independently in SSL, using intermediate layers of a neural network–instead of the deepest layer where the initial training criterion was applied– has long been known to be useful in transfer learning scenarios [232]. Features in upstream layers often appear more general and transferable to various downstream tasks than the ones at the deepest layers which are too specialized towards the initial training objective. This strongly suggests a related explanation for its success in SSL: does removing the last layers of a trained SSL model improve performances *because of a misalignment between the SSL training task (source domain) and downstream task (target domain)?*

In this paper, we examine that question thoroughly. We first place the SSL *trick of removing the projector post-training* under the umbrella of a generically applicable method that we call **Guillotine Regularization**. We argue that it is important to distinguish the action of removing layers during evaluation from architecture modifications because the optimal layer to use for a given downstream task is not always the backbone and could be intermediate projector's layers. Then, we explore how changes in the training optimization, training data and downstream task impact the optimal layer in both supervised and self-supervised setting. Lastly, we demonstrate that increasing the *alignment* between the pretext and downstream task in SSL decreases the need to use a projector in SSL.

To summarize, this paper's main contributions are the following:

- Since the optimal layer to use in Self-Supervised-Learning might not always be the backbone, we suggest coining the action of removing layer as a general method called Guillotine Regularization to distinguish it from the architectural modification which is the addition of a projector.



(a) (b)

Fig. 1. a) An illustration of projector head trick used in SSL. During training, a small neural network named the *Head* (also coined as *projector* in the SSL literature [49]) is added on top of another deep network refereed as the *Trunk*. This *Head* can be viewed as a buffer between the training loss and the Trunk that can absorb any bias related to a ill optimisation. When using such network on downstream tasks, we throw away the Head. b) We measure with linear probes the accuracy at different layers on a Resnet50 (as Trunk) (see Figure 3 for vision transformers) on which we added a small 3-layer MLP (as Head) for various supervised and self-supervised methods. For each method, we show the mean and standard deviation across 3 runs (The std between different runs is low). With traditional supervised learning, there is a significant drop in performances when using the trunk layer instead of the last projector layer. However, when looking at self-supervised methods, the gap in performances between the linear probe trained at the trunk and projector can be as high as 30%.

- To show through experiments that the optimal layer to cut heavily depend on the training optimization, training data and downstream task for both supervised and self-supervised models. We hope that this result will encourage the research community to run more systematic evaluations through different layers.
- The need to use Guillotine Regularization in SSL depends heavily on how the positives views are defined. When these views are aligned with the downstream-task, the optimal layer to use become closer to the last layer.

3.2. Related work

Self-supervised learning Many recent works on self-supervised learning [49, 52, 93, 134, 40, 236, 22, 199, 43, 53, 136, 247, 248] rely on the addition of few non linear layers (MLP) – termed *projection head* – on top of a well established neural network – termed *backbone* – during training. This addition is done regardless of the neural network used as backbone, it could be a ResNet50 [96] or a Vision Transformer [67]. After training, the projector is usually threw away to evaluate the model using the backbone representation. Even if Chen et al. [50] demonstrated that the optimal layer to use might not always be the backbone when using few labelled data, most recent works introducing new

SSL methods have continued to use only the backbone for evaluation. Some works also tried to understand why a projection head is needed for self-supervised learning. Appalaraju et al. [6] argue that the nonlinear projection head acts as filter that can separate the information used for the downstream task from the information useful for the contrastive loss. In order to support this claim, they used deep image prior [205] to perform features inversion to visualize the features at the backbone level and also at the projector level. They observe that features at the backbone level seem more suitable visually for a downstream classification task than the ones at the projector level. Another related work [29] similarly tries to map back the representations to the input space, this time by using a conditional diffusion generative model. The authors present visual evidence confirming that much of the information about a given input is lost at the projector level while most of it is still present at the backbone level. Another line of work tries to train self-supervised models without the use of a projector. Jing et al. [117] shows that by removing the projector and cutting the representation vector in two parts, such that a SSL criteria is applied on the first part of the vector while no criterion is applied on the second part, improves considerably the performances compared to applying the SSL criteria directly on the entire representation vector. This however works mostly thanks to the residual connection of the resnet. In contrast with these approaches, our work focus on identifying which components of traditional SSL training pipelines can explain why the performances when using the final layers of the network are so much worse than the ones at the backbone level. This identification will be key for designing future SSL setups in which the generalisation performance doesn't drop drastically when using the embedding that the SSL criterion actually learns.

Transfer learning The idea of using the intermediate layers of a neural network is very well known in the transfer learning community. Work like Deep Adaptation Network [144] freeze the first layers of a neural network, fine-tune the last layers while adding a head which is specific for each target domain. The justification behind this strategy is that deep networks learn general features [45, 24, 26], especially the ones at the first layers, that may be reused across different domain [232]. Oquab et al. [161] demonstrate that when limited amount of training data are available for the target tasks, using the frozen features extracted from the intermediate layers of a deep network trained on classification can help solve object and action classification tasks on other datasets. Another line of work on training with random or noisy labels also studied how the use of intermediate layers improves significantly downstream performances [149] while Baldock et al. [16] introduced a measure of example difficulty that leverages the number of intermediate layers that are aligned towards a given prediction. In this paper, we show that SSL trained models fall under the realm of transfer learning, in consequence we can expect that all the observations made in the transfer learning literature about the use of intermediate layers are also valid for SSL. When viewing the

projector SSL trick and cutting layer for transfer as a general machine learning trick to improve generalization, it's not surprising anymore that work as Wang et al. [219], Sarıyıldız et al. [182] have been able to show that adding a projector can also be highly beneficial for supervised training.

Out of distribution (OOD) generalization. Kirichenko et al. [123] demonstrates that retraining only the last layer with a specific reweighting helps to "forget" the spurious correlations that were learned during the training. Such work emphasizes that most of the spurious correlation due to the training objective is contained in the last layers of the network. Thus, retraining them is essential to remove such spurious correlation and generalize better on downstream tasks. Similarly Rosenfeld et al. [172] show that retraining only the last layers is most of the time as good as retraining the entire network over a subset of downstream tasks. Lastly, Evci et al. [77] demonstrates the usefulness of using intermediate layers for OOD. Our study also confirms that Guillotine Regularization show important properties with respect to OOD generalization.

3.3. Guillotine Regularization: A regularization scheme to improve generalization of deep networks

In this section, we provide a definition for Guillotine Regularization. Then, through experiments, we show that the optimal layer to use changes significantly depending on different factors. Finally, we show that the performances at a given layer are not always correlated with the performances one can have at another layer.

3.3.1. (Re)Introducing Guillotine Regularization From First Principles

We distinguish between a **source** *training task* with its associated training set, and a **target** *downstream task* with its associated dataset[1]. It is the performance on the downstream task that is ultimately of interest. In the simplest of cases both tasks could be the same, with their datasets sampled i.i.d. from the same distribution. But more generally they may differ, as in SSL or transfer learning scenarios. In SSL we typically have an *unsupervised* training task, that uses a training set with no labels, while the downstream task can be a supervised classification task. Also note that while the bulk of training the model's parameters happens with the training task, transferring to a different downstream task will require some additional, typically lighter, training, at least of a final layer specific for that task. In our study we will focus on the use of a representation computed by the network trained on the training task and then frozen, which gets fed to a simple linear layer that will

[1]Terminology pretext-training / downstream comes from SSL, while source / target is used in transfer learning

be tuned for the downstream task. This "linear evaluation" procedure is typical in SSL and aims to evaluate the quality/usefulness of an unsupervised-trained *representation*. Our focus is to ensure good generalization to the downstream task. Note that training and downstream tasks may be misaligned in several different ways.

Informally, Guillotine Regularization consists in the following: for the *downstream task*, rather than using the last layer (layer L) representation from the network trained on the *training task*, instead use the representation from a few layers above (layer t, with $t < L$). We thus *remove* a small multilayer "head" (layers $t+1$ to L) of the initially trained network, hence the name of the technique. We call the remaining part (layers 1 to t) the *trunk*[2].

Formally, we consider a deep network that takes an input X and computes a sequence of intermediate representations $H_1, \ldots, H_L$ through layer functions $f^{(1)}, \ldots f^{(L)}$ such that $H_\ell = f^{(\ell)}(H_{\ell-1})$, starting from $H_0 = X$. The entire computation from input X to last layer representation H_L is thus a composition of layer functions[3]:

$$H_L = f_{\theta,\phi}(X) = (\underbrace{f^{(L)} \circ \cdots \circ f^{(t+1)}}_{\text{head } f_\phi^{t+1:L}} \circ \underbrace{f^{(t)} \circ \cdots \circ f^{(1)}}_{\text{trunk } f_\theta^{1:t}})(X)$$

The parameters θ and ϕ of trunk $f_\theta^{1:t}$ and head $f_\phi^{t+1:L}$ are then trained on the entire training set of examples $\mathbf{X}^{\text{source}}$ of the training task (optionally with associated targets $\mathbf{Y}^{\text{source}}$ that we may have in transfer scenarios, but will typically be absent in SSL), to minimize the training task objective L^{source}:

$$\hat{\theta}, \hat{\phi} = \arg\min_{\theta,\phi} L^{\text{source}}(f_\phi^{t+1:L}(f_\theta^{1:t}(\mathbf{X}^{\text{source}})), \mathbf{Y}^{\text{source}})$$

Then the multilayer head $f_\phi^{t+1:L}$ is discarded, we add to the trunk a (usually shallow) new head s_w and we train its parameters w, using the training set of examples for the downstream task $(\mathbf{X}^{\text{target}}, \mathbf{Y}^{\text{target}})$, to minimize the downstream task objective L^{target}:

$$\hat{w} = \arg\min_{w} L^{\text{target}}(s_w(\underbrace{f_{\hat{\theta}}^{1:t}(\mathbf{X}^{\text{target}})}_{\text{representation } \mathbf{H}^{\text{target}}}), \mathbf{Y}^{\text{target}})$$

[2]head / trunk are also known as projection head / backbone in the SSL literature

[3]Precisely, a "layer function" $f^{(\ell)}$ can correspond to a standard neural network layer (fully-connected, convolutional) with no residual or shortcut connections between them, or to entire blocks (as in densenet, or transformers) which may have internal shortcut connections, but none between them.

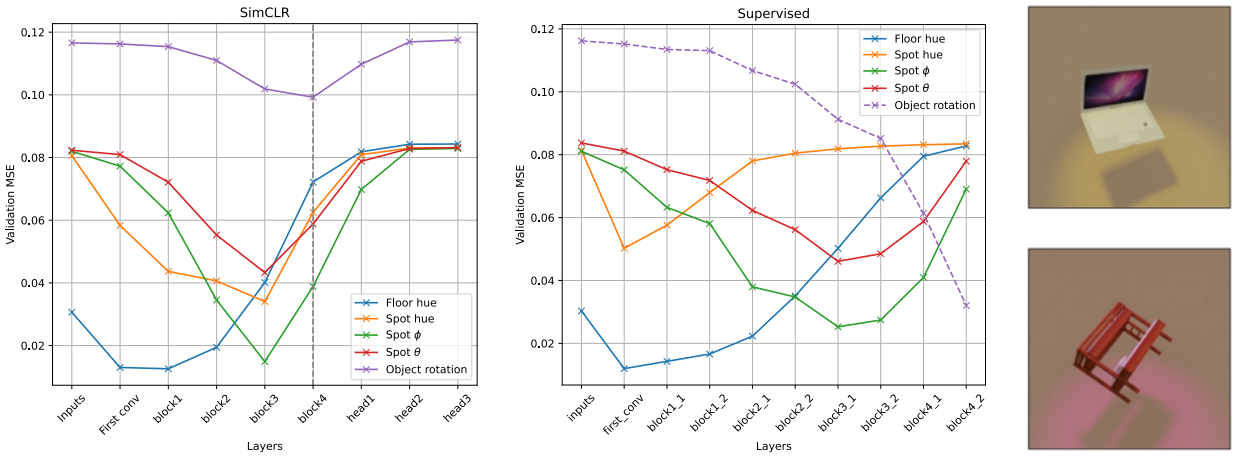


Fig. 2. Training a linear regression to predict latent variables from pooled intermediate representations of a network trained with a self-supervised objective (using SimCLR) or a supervised objective (trained to predict 3D rotations of an object). The data used consists of renderings of 3d objects from 3D Warehouse [202] where we control the floor, lighting and object pose with latent variables, see samples on the right. The dimension of the intermediate representations increases throughout the layers and is kept constant in the head, if there is one. In the supervised setting, when looking at the Validation Mean Squared Error for object rotations prediction, the lowest error is obtained with the linear probe at the last layer of the neural networks. In contrast, the lowest error for other attributes like the Spot θ prediction are obtained with the linear probes localized 3,4 or 5 layers before the output of the networks. In the self-supervised setting, we also see that the predictor is responsible for a lot of the invariance to augmentation, and that the information is most easily retrievable before it. These results highlight the need to use Guillotine Regularization i.e removing the last layers of the neural network to generalize better on other tasks.

3.3.2. An empirical analysis of situations in which cutting layers is beneficial

There are several situations that can create a misalignment between a training and a downstream task. Here we name of few:

Misalignment between the training (source) and downstream (target) task while using the same input data distribution. The potential effectiveness of GR for transfer is not surprising since this technique has been used for years in the transfer learning research literature [232] to improve generalization across different tasks. As a simple illustration, we present Figure 2 which show how much performances on a given task can vary depending on which layer has been chosen as features extractor. In this figure, we used an artificially created object dataset in which we are able to play with different factors of variations. The dataset consists of renderings of 3D models from 3D warehouse [202]. Each scene is built from a 3D object, a floor and a spot placed on top of the object to add lighting.

This allows us to control every factor of variation and produce complex transformations in the scene. We vary the rotation of the object defined as a quaternion, the hue of the floor, and the spot hue as well as it position on a sphere using spherical coordinates. We provide more details on the dataset and rendering samples in the appendix. We observe in Figure 2 that when training a supervised model on the object rotation prediction task and evaluating the linear probe on the same task across different layers, the best results are obtained on the last layer. However, when using the same frozen neural network to predict other attributes like the Spot θ, the best performances are obtained few layers before the last one. Similarly, when training with a self-supervised objective (SimCLR), we can see that the different factors of variation are most easily retrievable before the projector. This means that representations before the projector will be more versatile as they will contain information that was removed by the pretraining task. For example if our downstream task is to predict the rotation, the representation at block4 will be optimal while if the downstream task is to predict the spot hue, the representation at the block 3 will be optimal. Such results highlight the need to use Guillotine Regularization when there is a shift in the prediction task. Moreover, the optimality of a layer depends on the downstream task.

Misalignment due to badly optimized network It can be expected that the optimal layer to use to train a downstream task readout function might be different depending on how much the pretrained network is overfitting on the pretext task. To test this hypothesis, we train a headed supervised Resnet50 on ImageNet with two different types of optimization. The first one uses only AdamW with a small learning rate of $1e-4$ without any additional regularization. The second one uses SGD and the recommended hyper-parameters for supervised training (with cycling learning rate, weight decay, and momentum). In Figure 3a, we observe that the AdamW trained network that is overfitting on the classification task has readout function performances that are very close across different layers. However, when looking at the well-regularized model with SGD which does not overfit on the task, the readout performances across layers vary significantly. In a second experiment, we study more in-depth the effect of overfitting by training the Resnet50 over only a random subset of 250 classes. Then we use the remaining 750 classes as an OOD validation set that is split randomly in other subset of 250 classes. In Figure 3b, we clearly see that the training readout is overfitting on the training set while the readout performances across layers are similar on the

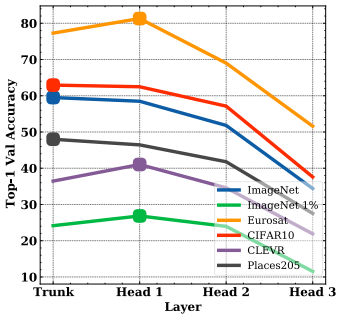
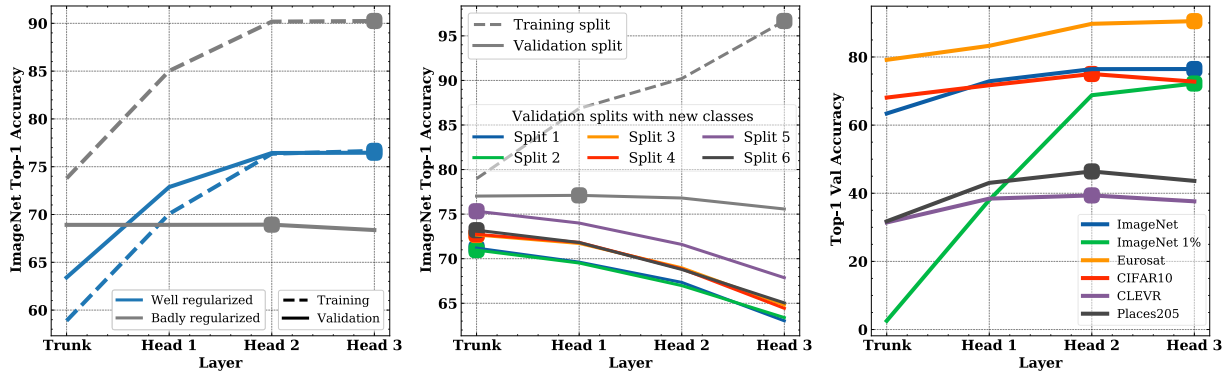


Fig. 4. SimCLR: Linear probe accuracy on several downstream tasks. **The optimal layer to cut is not the same for different downstream tasks.**



(a) Training setups **(b)** Random split **(c)** Downstream tasks

Fig. 3. Supervised: The optimal layer to cut might change depending of the training optimization, the data and the downstream task. The best accuracy for each curve is show as a big square. For each experiments, we trained a headed supervised Resnet50 over ImageNet (with a 3 layer MLP as projection head). For a) and c) we trained this network over the full training set whereas for b) we use a random subset of 250 classes. Then, we froze the model parameters and trained linear probes over representation at different layers. **a)** We trained two models with different optimization pipeline: the first one in blue was trained with SGD using a cycling learning rate, along with momentum and weight decay. The second one in gray was trained with AdamW without additional regularization. This model is overfitting on the training set, which leads to similar validation performances across the backbone and projector. In contrast, the first one generalize much better but the performances across layers change significantly. **b)** Validation accuracy given by linear probes on different random subset of 250 ImageNet's classes for each layers. The validation split in gray corresponds to the same subset of classes that was used for training whereas Split 1-6 corresponds to different OOD random split. In this instance, we see that the optimal layer to use is the first layer of the projector. **c)** Validation performances on different downstream tasks. We have used the well regularized model from a) and evaluate it across different downstream tasks. For some datasets, the optimal layer to use is the last one, while for some other the optimal layer is the second layer of the projector.

corresponding in distribution validation set (which is similar to the previous experiment over the full ImageNet). Then, we train linear probes over the OOD splits and observe that the performances are radically different from the in distribution validation set. In fact, in this instance the best layer to use for every of these split is the backbone layer whereas the best layer to use for the in-distribution split is the projector layer. **This result highlights that the optimal layer to discard can vary depending on the optimization techniques and downstream data distribution, even when the same training objective is used.**

Misalignment between the training and downstream tasks while using different data distributions. When using a pretrained model to predict new classes, there is a bias in the data distribution as well as in the fine-tuning objective (with respect to the training settings). We did a first experiment in Figure 3c in which we train a supervised

Resnet50 over ImageNet. Then, we freeze the weights of the model and train a linear probe over ImageNet [56], CIFAR10 [127], Place205 [246], CLEVR [118] and Eurosat [101] at different layers. We observe that the readout performances on ImageNet are the best at the last layer but for datasets like CLEVR or Place205 the best performances are obtained at the second projector layer. In Figure 4, we performed the same experiment but this time using SimCLR. In this instance, the best performances for ImageNet are obtained at the backbone whereas the best performances for Eurosat, CLEVR and Imagenet 1% are obtained at the first projector layer. **This result challenges the common practice of discarding the entire projector in SSL since the layers to cut depend on the downstream task.**

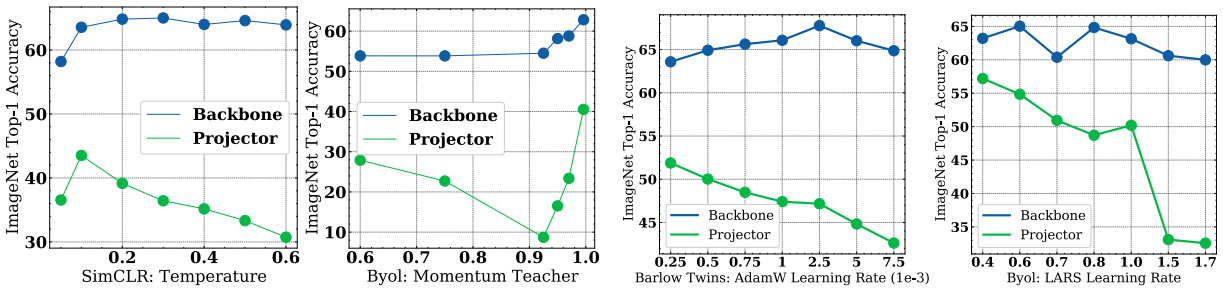
Misalignment between the training input data distribution and testing input data distribution while using the same training and downstream tasks. Another type of bias can arise when using a wrongful data distribution after training of the model. This scenario is often referred to Out Of Distribution (OOD) since the distribution of the data used by the model becomes different from the one seen during training. We took the supervised model trained on ImageNet along with the linear probe trained at different layers and evaluate the performances of these readouts on ImageNet-C [102] which is a modified version of the validation set of ImageNet on which different data transformations were applied. Our experiment in Table 1 demonstrates that the performances are better after cutting two layers from the head of the network which highlight that it might be a good practice to probe intermediate representations when evaluating on OOD tasks.

Head 3	Head 2	Head 1	Trunk
59.0	58.8	**58.0**	63.3

Table 1. ImageNet-C mCE (unormalized) across layers.

3.3.3. The readout performances at the projector and backbone level are not always correlated

In Figure 5 we study the effect of Guillotine Regularization with respect to an hyper-parameter grid search for various SSL methods (SimCLR, Barlow Twins and Byol). When looking at the performances on ImageNet using a linear probe at the backbone level, one can observe an almost stable classification task performance for different hyper-parameters such as SimCLR temperature, Barlow Twins and Byol learning rate while the corresponding performances at the projector level change significantly. This highlights that the performances at the projector level are not always correlated with the performances at the backbone level. In consequence, knowing the performances of a linear probe at the projector level cannot give in advance insights about the performances at the backbone level.



(a) SimCLR **(b)** Byol **(c)** Barlow Twins **(d)** Byol

Fig. 5. The performances at the projector level aren't always correlated with the performances at the backbone level. We train SimCLR, Barlow Twins and Byol with different hyper-parameters and evaluate with a linear prob, the performances at the backbone but also at the projector level on ImageNet classification task. For each model, we observe that the accuracy given by the linear probe at the backbone level isn't always correlated with the performance at the projector level.

3.4. Reducing the Need for a Projector in Self-Supervised Learning by increasing the alignment with the downstream task

Self-Supervised Learning is often considered a distinct learning paradigm in between supervised and unsupervised learning. In reality, the distinction is not as sharp, and much of SSL can be understood as solving a pretext-tasks akin to a supervised task[224, 120], merely with pseudo-labels obtained in another way than by human annotation. In this section, we show that different data selection process in SSL influences the alignment between the downstream and pretext task, which heavily impact the need of using a projector head in SSL.

To confirm the hypotheses that SSL methods need to use a projector because of a misalignement between the pretext and downstream task, we have to verify that reducing this misalignement, results in reducing the performance gap between the Trunk and Head representations. Ideally, we would like to get close to the supervised scenario in Figure 1 for which the optimal readout function is obtained at the last layer. To do so, we devise two experimental setups in which we replace the traditional data augmentation pipeline used in SSL, which consists of using handcrafted augmentation on each image to create a set of pairwise positive samples.

In the first setup, while using the exact same SSL criterion (SimCLR), we use as positive examples pairs of images that belong to the same class, and as negative examples images that don't belong to the same class. Note that the SSL training criteria will push towards a collapse in the representation space of all the images

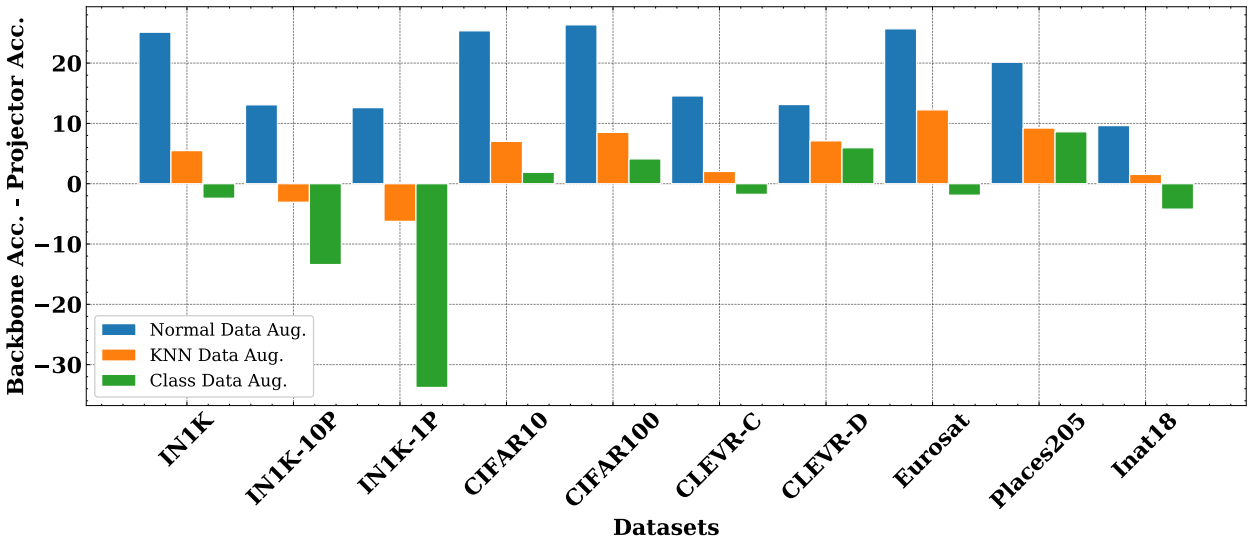
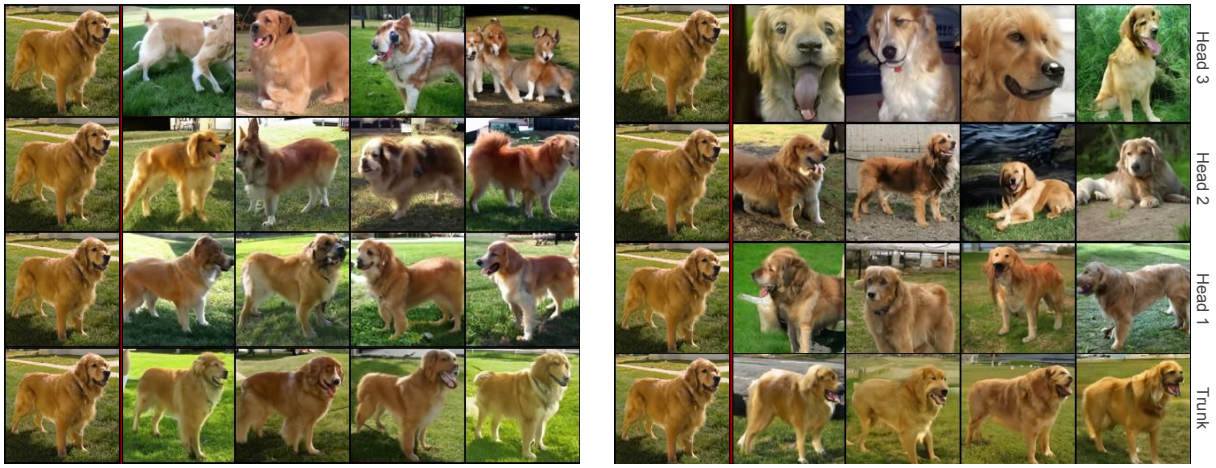


Fig. 6. Difference in accuracy with linear probing between the projector and backbone representation with different alignments with respect to the classification downstream task. In this experiment we used SimCLR and we change how the positive pair are defined to better aligned with a classification downstream task. In blue, our baseline, we trained SimCLR with the traditional SSL data augmentations which defines the positive view as two augmentations of a same image. In orange, we use the embedding of a pretrained model to define the positive pair as two nearest neighbors under a pretrained model (while using the same data augmentation as the baseline). In green, we use a supervised class label selection to define the positive examples. In this scenario, SimCLR should learn to produces similar embedding to all images belonging to a given class. All three models are trained on ImageNet (IN1K), then we evaluate them with a linear probe across a wide range of downstream tasks at the backbone and projector level and show the difference in accuracy between both. **When the difference is positive, the accuracy at the backbone level is higher than the one at the projector level, highlighting the benefits of Guillotine Regularization. In contrast when the difference is negative, the accuracy at the projector level is higher than the one at the backbone level. In this instance, Guillotine Regularization is not needed.** When positives pairs are defined as belonging to a given class, there is no misalignment with the imagenet classification downstream task. Thus on ImageNet-1K, ImageNet1k-10P (10% of the training set to train the linear probe) and ImageNet1k-1P (1% of the training set to train the linear probe), we observe that the performances at the projector level are much higher than the ones at the backbone level. Interestingly, the nearest neighbors heuristic reduces considerably the impact of Guillotine Regularization across several downstream tasks.

belonging to the same class, while pushing further apart the different class clusters. By doing so the training SSL objective becomes perfectly aligned with the downstream classification task, despite using a SSL training criteria instead of a traditional cross entropy loss.

In the second setup, we use as positive pairs the closest neighbors found by a pretrained SSL model trained with the traditional SSL handcrafted data augmentation pipeline. The reasoning is that if instead of considering each image of the dataset as its own specific class, we use clusters of many images to define the positive pairs, we



(a) SimCLR trained with SSL augmentations. **(b)** SimCLR trained with class labels.

Fig. 7. In this figure, we used RCDM [31], a conditional generative model to visualize what information is decodable at different layers. The leftmost column of images (before the red line) is the conditioning image that was used to compute the representation that is fed to RCDM. The subsequent columns are samples generated by the model using this representation. The first row correspond to the last projector layer, the second row to the second projector layer, the third one to the first projector layer and the last row to the backbone layer. As show in Figure 6, changing the alignment with the pretext task change significantly the information encoded by the neural network. When using SSL augmentations at the projector level, the information about the dog's breed seem to have been lost whereas when looking at a network trained with supervised augmentations, the information is preserved throughout each layers.

might be able to close the gap with respect to a supervised baseline without the need of labels.

In Figure 6, we show the differences in accuracy between the backbone and the projector with respect to these two new data augmentation scenarios. The baseline, using the traditional SimCLR positive pairs based on data augmentations is in blue, the nearest neighbors setup in orange and the class based setup in green. We observe for SimCLR that using the nearest neighbors based heuristic is helping in reducing the gap between the pretext and downstream task while having a purely supervised heuristic to define the positive pair is removing the need to perform Guillotine Regularization across several downstream tasks. Hence confirming the hypothesis that the effectiveness of a projector depends of the alignment between the pretext and downstream task in self-supervised learning.

3.4.1. Visualizing the information across layers for different alignments

In this section, we use RCDM [31], a conditional generative model to visualize what information is retain or not in the representation. We train RCDM on ImageNet with blurred

faces[229], using the representation given by a SimCLR model trained on handcrafted SSL views and another which was trained on class based views. In Figure 7, we show that when looking at different decoding corresponding to different layers in the network, the information encoded vary a lot depending on the layer to use. When going deeper, RCDM is not able to reconstruct as much as information about the images than when using the backbone representation (which contain much more low level features). When looking at the generated samples that were conditioned on the representation of the model trained with supervised views, we observe that the breed of the dog stay the same across layers. However when using traditional data augmentations, the information about the specific golden retriever breed is lost in the last projector layers. This is correlated with the fact that this model get lower classification performances when using the projector.

3.4.2. Experimental details

We use Pytorch [163] and FFCV-SSL [32, 129] as data loader. All the experiments were performed with a Resnet50 [96] (except if mentioned otherwise) as backbone. For each model, we use a batch of size 2048 and AdamW [145] as optimizer with an adaptive learning rate schedule. We run the training for 100 epochs. For each model, we add as head a small MLP of 3 layers of size 2048 (same dimension as the backbone) with ReLU [86] as activation and batch normalization [113]. When training different SSL methods, we always used the same set of data augmentations (with cropping, color-jitter, random grayscale, gaussian blur and solarization).

3.5. Conclusion

Through empirical evaluations, we demonstrated that the optimal layer to use for downstream evaluation vary depending on several factors: optimization, data and downstream task. These results highlight the need for SSL practitioners to run systematic evaluations at several layers instead of using always the backbone as reference. We also demonstrated that the use of a projector in SSL depends on the alignment between the downstream and pretext task. Despite, its usefulness, having to rely on a *trick* like Guillotine Regularization to increase performances reveals an important shortcoming of current self-supervised learning methods: the inability to design experimental setups and training criteria that learn structured and truly invariant representations with respect to an appropriate set of factors of variation. As future work, in order to escape from Guillotine Regularization, we should focus on finding new training criteria and data augmentations that will be more *aligned* with the downstream tasks of interest.

Prologue to Article 3

Do SSL Models Have Déjà Vu? A Case of Unintended Memorization in Self-supervised Learning, *Casey Meehan*, Florian Bordes*, Pascal Vincent, Kamalika Chaudhuri†, Chuan Guo†*, Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS 2023) [4]

Kamalika and Chuan wanted to explore the use of conditional generative models (like RCDM) for assessing how much SSL methods are memorizing their training data. Casey, advised by Kamalika and Chuan, started to work on this project for his internship at Meta. Casey asked me and Pascal to provide guidance and feedback concerning the use of RCDM to reconstruct the training data memorized by SSL methods. The main hypothesis behind this project was that SSL representations are much richer than supervised representations. In consequence, it might be easier to extract private information from a pre-trained SSL model than a supervised one. In this article, we showed that indeed some SSL methods are extremely vulnerable to specific attacks and that it is even possible to reconstruct extremely specific information about the training data.

Contribution statement The early phase of this project was done by Casey during his internship while I had a more advisory role concerning the use of RCDM. However, after Casey's internship ended, I took the lead role in the project by running additional experiments to strengthen our contributions. I contributed to having a fine-grained analysis of memorization with respect to different SSL criteria, Guillotine regularization, number of parameters, and architecture. Lastly, I suggested and implemented the Déjà Vu score to have a single number to measure memorization in SSL models. I also wrote a significant part of the NeurIPS rebuttal and added the fine-tuning experiments.

[4] * denotes equal contribution and † denotes equal direction contribution.

Chapter 4

Article 3: Do SSL Models Have Déjà Vu? A Case of Unintended Memorization in Self-supervised Learning

Self-supervised learning (SSL) algorithms can produce useful image representations by learning to associate different parts of natural images with one another. However, when taken to the extreme, SSL models can unintendedly memorize *specific* parts in individual training samples rather than learning semantically meaningful associations. In this work, we perform a systematic study of the unintended memorization of image-specific information in SSL models—which we refer to as *déjà vu memorization*. Concretely, we show that given the trained model and a crop of a training image containing only the background (*e.g.*, water, sky, grass), it is possible to infer the foreground object with high accuracy or even visually reconstruct it. Furthermore, we show that *déjà vu* memorization is common to different SSL algorithms, is exacerbated by certain hyperparameter choices, and cannot be detected by conventional techniques for evaluating representation quality. Our study of *déjà vu* memorization reveals previously unknown privacy risks in SSL models, as well as suggests potential practical mitigation strategies.

4.1. Introduction

Self-supervised learning (SSL) [49, 51, 236, 23, 40, 98] aims to learn general representations of content-rich data without explicit labels by solving a *pretext task*. In many recent works, such pretext tasks rely on joint-embedding architectures whereby randomized image augmentations are applied to create multiple views of a training sample, and the model is trained to produce similar representations for those views. When using cropping as random image augmentation, the model learns to associate objects or parts (including the background scenery) that co-occur in an image. However, doing so also arguably exposes the training data to higher privacy risk as objects in training images can be explicitly memorized by the

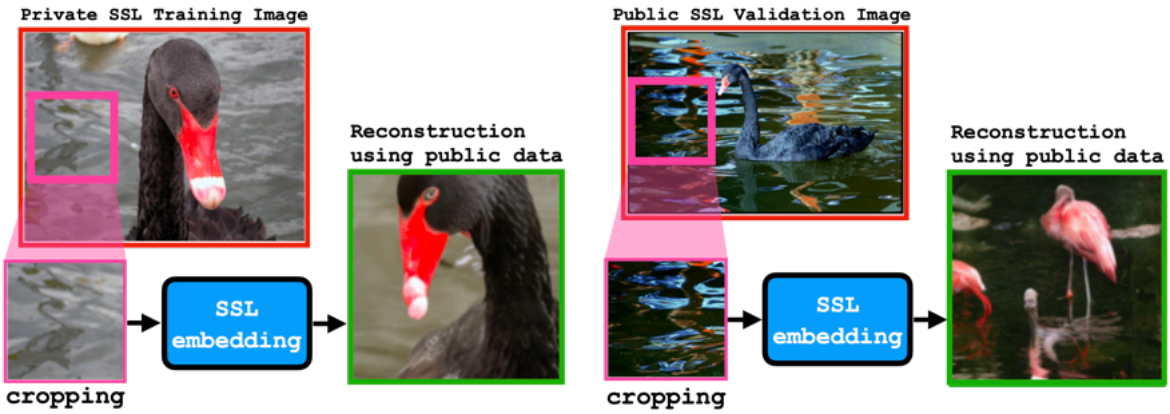


Fig. 1. Left: Reconstruction of an SSL training image from a crop containing only the background. The SSL model memorizes the association of this *specific* patch of water (pink square) to this *specific* foreground object (a black swan) in its embedding, which we decode to visualize the full training image. **Right:** The reconstruction technique fails on a public test image that the SSL model has not seen before.

SSL model. This may allow an adversary to extract such information from the trained model for targeted individuals.

In this work, we aim to evaluate to what extent SSL models memorize the association of specific objects in training images or the association of objects and their specific backgrounds, and whether this memorization signal can be used to reconstruct the model's training samples. Our results demonstrate that SSL models memorize such associations beyond simple correlation. For instance, in Figure 1 (**left**), we use the SSL representation of a *training image crop containing only water* and this enables us to reconstruct the object in the foreground with remarkable specificity—in this case a black swan. By contrast, in Figure 1 (**right**), when using the *crop from the background of a test set image* that the SSL model *has not seen before*, its representation only contains enough information to infer, through correlation, that the foreground object was likely some kind of waterbird — but not the specific one in the image.

Figure 1 shows that SSL models suffer from the unintended memorization of images in their training data—a phenomenon we refer to as *déjà vu memorization* [1] Beyond visualizing *déjà vu* memorization through data reconstruction, we also design a series of experiments to quantify the degree of memorization for different SSL algorithms, model architectures, training set size, *etc.* We observe that *déjà vu* memorization is exacerbated by the atypically large number of training epochs often recommended in SSL training, as well as certain hyperparameters in the SSL training objective. Perhaps surprisingly, we show that *déjà vu* memorization occurs even when the training set is large—as large as half of

[1]The French loanword *déjà vu* means 'already-seen', just as an image is seen and memorized in training.

ImageNet [57]—and can continually worsen even when standard techniques for evaluating learned representation quality (such as linear probing) do not suggest increased overfitting. Our work serves as the first systematic study of unintended memorization in SSL models and motivates future work on understanding and preventing this behavior. Specifically, we:

- Elucidate how SSL representations memorize aspects of individual training images, what we call *déjà vu* memorization;
- Design a novel training data reconstruction pipeline for non-generative vision models. This is in contrast to many prominent reconstruction algorithms like [37, 38], which rely on the model itself to generate its own memorized samples and is not possible for SSL models or classifiers;
- Propose metrics to quantify the degree of *déjà vu* memorization committed by an SSL model. This allows us to observe how *déjà vu* changes with training epochs, dataset size, training criteria, model architecture and more.

4.2. Preliminaries and Related Work

Self-supervised learning (SSL) is a machine learning paradigm that leverages unlabeled data to learn representations. Many SSL algorithms rely on *joint-embedding* architectures (*e.g.*, SimCLR [49], Barlow Twins [236], VICReg [23] and Dino [42]), which are trained to associate different augmented views of a given image. For example, in SimCLR, given a set of images $\mathcal{A} = \{A_1, \ldots, A_n\}$ and a randomized augmentation function aug, the model is trained to maximize the cosine similarity of draws of $\text{SSL}(\text{aug}(A_i))$ with each other and minimize their similarity with $\text{SSL}(\text{aug}(A_j))$ for $i \neq j$. The augmentation function aug typically consists of operations such as cropping, horizontal flipping, and color transformations to create different views that preserve an image's semantic properties.

SSL representations. Once an SSL model is trained, its learned representation can be transferred to different downstream tasks. This is often done by extracting the representation of an image from the *backbone model*[2] and either training a linear probe on top of this representation or finetuning the backbone model with a task-specific head [30]. It has been shown that SSL representations encode richer visual details about input images than supervised models do [29]. However, from a privacy perspective, this may be a cause for concern as the model also has more potential to overfit and memorize precise details about the training data compared to supervised learning. We show concretely that this privacy risk

[2]SSL methods often use a trick called *guillotine regularization* [30], which decomposes the model into two parts: a *backbone model* and a *projector* consisting of a few fully-connected layers. Such trick is needed to handle the misalignment between the pretext SSL task and the downstream task.

can indeed be realized by defining and measuring *déjà vu* memorization.

Privacy risks in ML. When a model is overfit on privacy-sensitive data, it memorizes specific information about its training examples, allowing an adversary with access to the model to learn private information [231, 78]. Privacy attacks in ML range from the simplest and best-studied *membership inference attacks* [186, 178, 175] to *attribute inference* [79, 151, 116] and *data reconstruction* [37, 18, 94] attacks. In the former, the adversary only infers whether an individual participated in the training set. Our study of *déjà vu* memorization is most similar to the latter: we leverage SSL representations of the training image background to infer and reconstruct the foreground object. In another line of work in the NLP domain [36, 37]: when prompted with a context string present in the training data, a large language model is shown to generate the remainder of string at test time, revealing sensitive text like home addresses. This method was recently extended to extract memorized images from Stable Diffusion [38]. We exploit memorization in a similar manner: given partial information about a training sample, the model is prompted to reveal the rest of the sample.[3] In our case, however, since the SSL model is not generative, extraction is significantly harder and requires careful design.

4.3. Defining *Déjà Vu* Memorization

What is *déjà vu* memorization? At a high level, the objective of SSL is to learn general representations of objects that occur in nature. This is often accomplished by associating different parts of an image with one another in the learned embedding. Returning to our example in Figure 1, given an image whose background contains a patch of water, the model may learn that the foreground object is a water animal such as duck, pelican, otter, *etc.*, by observing different images that contain water from the training set. We refer to this type of learning as *correlation*: the association of objects that tend to co-occur in images from the training data distribution.

A natural question to ask is *"Can the reconstruction of the black swan in Figure 1 be reasoned as correlation?"* The intuitive answer may be no, since the reconstructed image is qualitatively very similar to the original image. However, this reasoning implicitly assumes that for a random image from the training data distribution containing a patch of water, the foreground object is unlikely to be a black swan. Mathematically, if we denote by $\mathcal{P}$ the

[3]We recognize that it is easier to find a context string that might have been in the training data of a large language model than to find the exact pixels that constitute a crop of a training image. However, this paper focuses on revealing a memorization phenomenon in SSL and does not aim to provide a complete picture of all the privacy risks that it might entail.

training data distribution and A the image, then

$$p_{\text{corr}} := \mathbb{P}_{A\sim\mathcal{P}}(\text{object}(A) = \texttt{black swan} \mid \text{crop}(A) = \texttt{water})$$

is the probability of inferring that the foreground object is a black swan through *correlation*. This probability may be naturally high due to biases in the distribution $\mathcal{P}$, *e.g.*, if $\mathcal{P}$ contains no other water animal except for black swans. In fact, such correlations are often exploited to learn a model for image inpainting with great success [234, 204].

Despite this, we argue that reconstruction of the black swan in Figure 1 is *not* due to correlation, but rather due to *unintended memorization*: the association of objects unique to a single training image. As we will show in the following sections, the example in Figure 1 is not a rare success case and can be replicated across many training samples. More importantly, failure to reconstruct the foreground object in Figure 1 (**right**) on test images hints at inferring through correlation is unlikely to succeed—a fact that we verify quantitatively in Section 4.4.1. Motivated by this discussion, we give a verbal definition of *déjà vu* memorization below, and design a testing methodology to quantify *déjà vu* memorization in Section 4.3.1.

Definition: A model exhibits *déjà vu memorization* when it retains information so specific to an individual training image, that it enables recovery of aspects particular to that image given a part that does not contain them. The recovered aspect must be beyond what can be inferred using only correlations in the data distribution.

We intentionally kept the above definition broad enough to encompass different types of information that can be inferred about the training image, including but not restricted to object category, shape, color and position. For example, if one can infer that the foreground object is red given the background patch with accuracy significantly beyond correlation, we consider this an instance of *déjà vu* memorization as well. We mainly focus on object category to quantify *déjà vu* memorization in Section 4.4 since the ground truth label can be easily obtained. We consider other types of information more qualitatively in the visual reconstruction experiments in Section 4.5.

Distinguishing memorization from correlation. When measuring *déjà vu* memorization, it is crucial to differentiate what the model associates through *memorization* and what it associates through *correlation*. Our testing methodology is based on the following intuitive definition.

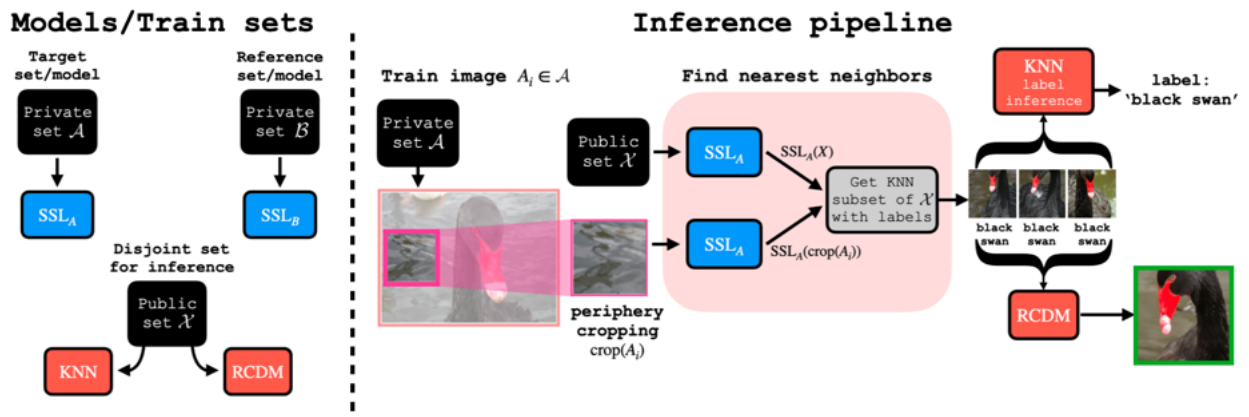


Fig. 2. Overview of testing methodology. **Left:** Data is split into *target set* $\mathcal{A}$, *reference set* $\mathcal{B}$ and *public set* $\mathcal{X}$ that are pairwise disjoint. $\mathcal{A}$ and $\mathcal{B}$ are used to train two SSL models SSL_A and SSL_B in the same manner. $\mathcal{X}$ is used for KNN decoding or for training an RCDM to reconstruct the input at test time. **Right:** Given a training image $A_i \in \mathcal{A}$, we use SSL_A to embed $\text{crop}(A_i)$ containing only the background, as well as the entire set $\mathcal{X}$ and find the k-nearest neighbors of $\text{crop}(A_i)$ in $\mathcal{X}$ in the embedding space. These KNN samples can be used directly to infer the foreground object (*i.e.*, class label) in A_i using a KNN classifier, or their embeddings can be averaged as input to the trained RCDM to visually reconstruct the image A_i. For instance, the RCDM reconstruction results in Figure 1 (left) when given $\text{SSL}_A(\text{crop}(A_i))$ and results in Figure 1 (right) when given $\text{SSL}_A(\text{crop}(B_i))$ for an image $B_i \in \mathcal{B}$.

Definition: If an SSL model associates two parts in a training image, we say that it is due to *correlation* if other SSL models trained on a similar dataset from $\mathcal{P}$ without this image would likely make the same association. Otherwise, we say that it is due to *memorization*.

Notably, such intuition forms the basis for differential privacy (DP; Dwork et al. [70], Dwork and Roth [69])—the most widely accepted notion of privacy in ML.

4.3.1. Testing Methodology for Measuring *Déjà Vu* Memorization

In this section, we use the above intuition to measure the extent of *déjà vu* memorization in SSL. Figure 2 gives an overview of our testing methodology.

Dataset splitting. We focus on testing *déjà vu* memorization for SSL models trained on the ImageNet-1K dataset [57]. Our test first splits the ImageNet training set into three independent and disjoint subsets $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{X}$. The dataset $\mathcal{A}$ is called the *target set* and $\mathcal{B}$ is called the *reference set*. The two datasets are used to train two separate SSL models, SSL_A and SSL_B, called the *target model* and the *reference model*. Finally, the dataset set $\mathcal{X}$ is used as an auxiliary public dataset to extract information from SSL_A and SSL_B. Our dataset splitting serves the purpose of distinguishing memorization from correlation in the following manner. Given a sample $A_i \in \mathcal{A}$, if our test returns the same result on SSL_A

and SSL_B then it is likely due to correlation because A_i is not a training sample for SSL_B. Otherwise, because $\mathcal{A}$ and $\mathcal{B}$ are drawn from the same underlying distribution, our test must have inferred some information unique to A_i due to memorization. Thus, by comparing the difference in the test results for SSL_A and SSL_B, we can measure the degree of *déjà vu* memorization[4].

Extracting foreground and background crops. Our testing methodology aims at measuring what can be inferred about the foreground object in an ImageNet sample given a background crop. This is made possible because ImageNet provides bounding box annotations for a subset of its training images—around 150K out of 1.3M samples. We split these annotated images equally between $\mathcal{A}$ and $\mathcal{B}$. Given an annotated image A_i, we treat everything inside the bounding box as the foreground object associated with the image label, denoted $\text{object}(A_i)$. We take the largest possible crop that does not intersect with any bounding box as the background crop (or *periphery crop*), denoted $\text{crop}(A_i)$[5]

KNN-based test design. Joint-embedding SSL approaches encourage the embeddings of random crops of a training image $A_i \in \mathcal{A}$ to be similar. Intuitively, if the model exhibits *déjà vu* memorization, it is reasonable to expect that the embedding of $\text{crop}(A_i)$ is similar to that of $\text{object}(A_i)$ since both crops are from the same training image. In other words, $\text{SSL}_A(\text{crop}(A_i))$ encodes information about $\text{object}(A_i)$ that cannot be inferred through correlation. However, decoding such information is challenging as these approaches do not learn a decoder associated with the encoder SSL_A.

Here, we leverage the public set $\mathcal{X}$ to decode the information contained in $\text{crop}(A_i)$ about $\text{object}(A_i)$. More specifically, we map images in $\mathcal{X}$ to their embeddings using SSL_A and extract the k-nearest-neighbor (KNN) subset of $\text{SSL}_A(\text{crop}(A_i))$ in $\mathcal{X}$. We can then decode the information contained in $\text{crop}(A_i)$ in one of two ways:

- *Label inference:* Since $\mathcal{X}$ is a subset of ImageNet, each embedding in the KNN subset is associated with a class label. If $\text{crop}(A_i)$ encodes information about the foreground object, its embedding will be close to samples in $\mathcal{X}$ that have the same class label (*i.e.*, foreground object category). We can then use a KNN classifier to infer the foreground object in A_i given $\text{crop}(A_i)$.
- *Visual reconstruction:* Following Bordes et al. [29], we train an RCDM—a conditional generative model—on $\mathcal{X}$ to decode SSL_A embeddings into images. The RCDM reconstruction

[4]See Appendix C.2.1 for details on how the dataset splits are generated.
[5]We also present another heuristic in appendix C.5 which takes a corner crop as the background crop, allowing our test to be run without bounding box annotations.

can recover qualitative aspects of an image remarkably well, such as recovering object color or spatial orientation using its SSL embedding. Given the KNN subset, we average their SSL embeddings and use the trained RCDM model to visually reconstruct A_i.

In Section 4.4, we focus on quantitatively measuring *déjà vu* memorization with label inference, and then use the RCDM reconstruction to visualize *déjà vu* memorization in Section 4.5.

4.4. Quantifying *Déjà Vu* Memorization

We apply our testing methodology to quantify a specific form of *déjà vu* memorization: inferring the foreground object (class label) given a crop of the background.

Extracting model embeddings. We test *déjà vu* memorization on a variety of popular SSL algorithms, with a focus on VICReg [23]. These algorithms produce two embeddings given an input image: a *backbone* embedding and a *projector* embedding that is derived by applying a small fully-connected network on top of the backbone embedding. Unless otherwise noted, all SSL embeddings refer to the projector embedding. To understand whether *déjà vu* memorization is particular to SSL, we also evaluate embeddings produced by a supervised model CLF_A trained on $\mathcal{A}$. We apply the same set of image augmentations as those used in SSL and train CLF_A using the cross-entropy loss to predict ground truth labels.

Identifying the most memorized samples. Prior works have shown that certain training samples can be identified as more prone to memorization than others [78, 220, 230]. Similarly, we provide a heuristic to identify the most memorized samples in our label inference test using confidence of the KNN prediction. Given a periphery crop, $\text{crop}(A_i)$, let $\text{KNN}_A\Big(\text{crop}(A_i)\Big) \subseteq \mathcal{X}$ denote its k-nearest neighbors in the embedding space of SSL_A. From this KNN subset we can obtain: **(1)** $\text{KNN}_A^{\text{prob}}\Big(\text{crop}(A_i)\Big)$, the vector of class probabilities (normalized counts) induced by the KNN subset, and **(2)** $\text{KNN}_A^{\text{conf}}\Big(\text{crop}(A_i)\Big)$, the negative entropy of the probability vector $\text{KNN}_A^{\text{prob}}\Big(\text{crop}(A_i)\Big)$, as confidence of the KNN prediction. When entropy is low, the neighbors agree on the class of A_i and hence confidence is high.

We can sort the confidence score $\text{KNN}_A^{\text{conf}}\Big(\text{crop}(A_i)\Big)$ across samples A_i in decreasing order to identify the most confidently predicted samples, which likely correspond to the most memorized samples when $A_i \in \mathcal{A}$.

4.4.1. Population-level Memorization

Our first measure of *déjà vu* memorization is population-level label inference accuracy: *What is the average label inference accuracy over a subset of SSL training images given their periphery crops?* To understand how much of this accuracy is due to SSL_A's *déjà vu* memorization, we compare with a correlation baseline using the reference model: KNN_B's label inference accuracy on images $A_i \in \mathcal{A}$. In principle, this inference accuracy should be significantly above chance level (1/1000 for ImageNet) because the periphery crop may be highly indicative of the foreground object through correlation, *e.g.*, if the periphery crop is a basketball player then the foreground object is likely a basketball. Figure 3 compares the accuracy of KNN_A to that of KNN_B when inferring the labels of images in $A_i \in \mathcal{A}$[6] using $\text{crop}(A_i)$. Results are shown for VICReg and the supervised model; trends for other models are shown in Appendix C.3.3. For both VICReg and supervised models, inferring the class of $\text{crop}(A_i)$ using KNN_B (dashed line) through correlation achieves a reasonable accuracy that is significantly above chance level.

However, for VICReg, the inference accuracy using KNN_A (solid red line) is significantly higher, and the accuracy gap between KNN_A and KNN_B indicates the degree of *déjà vu* memorization. We highlight two observations:

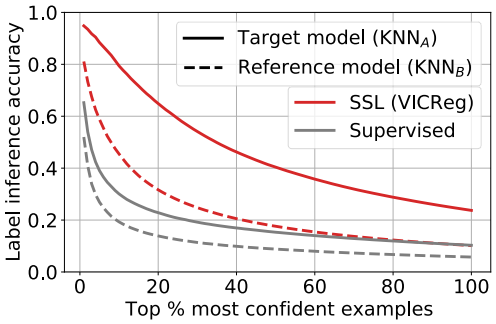


Fig. 3. Accuracy of label inference using the target model (trained on $\mathcal{A}$) vs. the reference model (trained on $\mathcal{B}$) on the top % most confident examples $A_i \in \mathcal{A}$ using only $\text{crop}(A_i)$. For VICReg, there is a large accuracy gap between the two models, indicating a significant degree of *déjà vu* memorization.

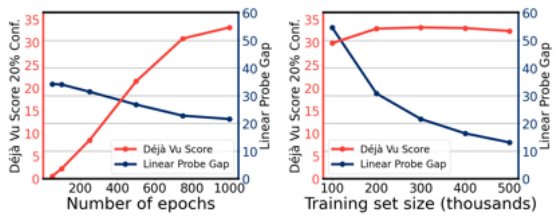
- The accuracy gap of VICReg is significantly larger than that of the supervised model. This is especially notable when accounting for the fact that the supervised model is trained to associate randomly augmented crops of images with their ground truth labels. In contrast, VICReg has no label access during training but the embedding of a periphery crop can still encode the image label.
- For VICReg, inference accuracy on the 1% most confident examples is nearly 95%, which shows that our simple confidence heuristic can effectively identify the most memorized samples. This result suggests that an adversary can use this heuristic to identify vulnerable training samples to launch a more focused privacy attack.

The *déjà vu* score. The curves of Figure 3 show memorization across confidence values for a single training scenario. To study how memorization changes with different

[6]The sets $\mathcal{A}$ and $\mathcal{B}$ are exchangeable, and in practice we repeat this test on images from $\mathcal{B}$ using SSL_B as the target model and SSL_A as the reference model, and average the two sets of results.

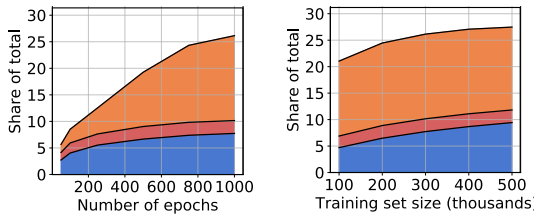
hyperparamters, we extract a single value from these curves: the *déjà vu score* at confidence level p. In Figure 3, this is the gap between the solid red (or gray) and dashed red (or gray) where confidence (x-axis) equal $p\%$. In other words, given the periphery crops of set $\mathcal{A}$, KNN_A and KNN_B separately select and label their top $p\%$ most confident examples, and we report the difference in their accuracy. The *déjà vu* score captures both the degree of memorization by the accuracy gap and the *ability to identify memorized examples* by the confidence level. If the score is 10% for $p = 33\%$, KNN_A has 10% higher accuracy on its most confident third of $\mathcal{A}$ than KNN_B does on its most confident third. In the following, we set $p = 20\%$, approximately the largest gap for VICReg (red lines) in Figure 3.

Comparison with the linear probe train-test gap. A standard method for measuring SSL performance is to train a linear classifier—what we call a 'linear probe'—on its embeddings and compute its performance on a held out test set. From a learning theory standpoint, one might expect the linear probe's train-test accuracy gap to be indicative of memorization: the more a model overfits, the larger is the difference between train set and test set accuracy. However, as seen in Figure 4, the linear probe gap (dark blue) fails to reveal memorization captured by the *déjà vu* score (red) [7].



(a) *déjà vu* vs. epochs **(b)** *déjà vu* vs. train set size

Fig. 4. Effect of training epochs and train set size with VICReg on *déjà vu* score (red) in comparison with linear probe accuracy train-test gap (dark blue). **Left:** *déjà vu* score increases with training epochs, indicating growing memorization while the linear probe baseline decreases significantly. **Right:** *déjà vu* score stays roughly constant with training set size suggesting that memorization may be problematic even for large datasets.



(a) *déjà vu* vs. epochs **(b)** *déjà vu* vs. train set size

Fig. 5. Partition of samples $A_i \in \mathcal{A}$ into the four categories: unassociated (not shown), memorized, misrepresented and correlated for VICReg. The memorized samples—those whose labels are predicted by KNN_A but not by KNN_B—occupy a significantly larger share of the training set than the misrepresented samples—those predicted by KNN_B but not KNN_A by chance.

[7] See section 4.6 for further discussion of the *déjà vu* score trends of Figure 4.

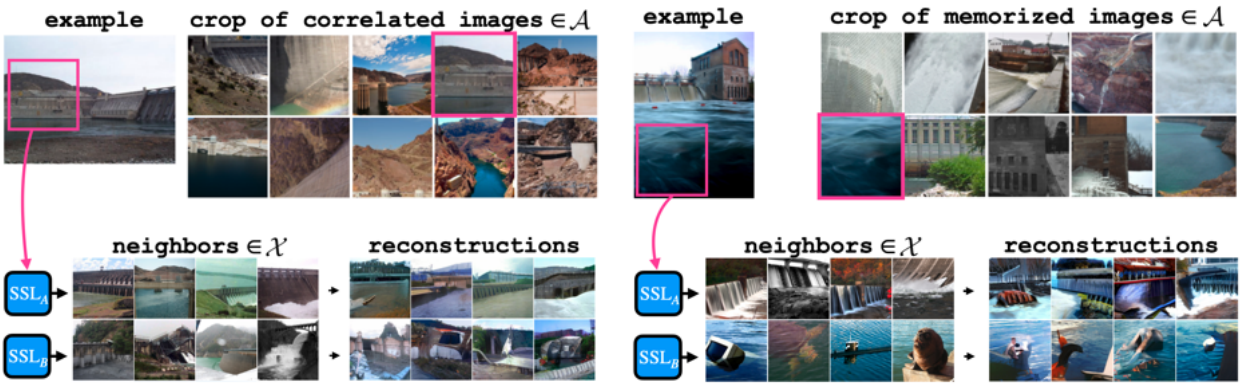
4.4.2. Sample-level Memorization

The *déjà vu* score shows, *on average*, how much better an adversary can select and classify images when using the target model trained on them. This average score does not tell us how many individual images have their label successfully recovered by KNN_A but not by KNN_B. In other words, how many images are exposed by virtue of *being in training set* $\mathcal{A}$: a risk notion foundational to differential privacy. To better quantify what fraction of the dataset is at risk, we perform a sample-level analysis by fixing a sample $A_i \in \mathcal{A}$ and observing the label inference result of KNN_A vs. KNN_B. To this end, we partition samples $A_i \in \mathcal{A}$ based on the result of label inference into four distinct categories: **Unassociated** - label inferred with neither KNN; **Memorized** - label inferred only with KNN_A; **Misrepresented** - label inferred only with KNN_B; **Correlated** - label inferred with both KNNs.

Intuitively, unassociated samples are ones where the embedding of $\text{crop}(A_i)$ does not encode information about the label. Correlated samples are ones where the label can be inferred from $\text{crop}(A_i)$ using correlation, *e.g.*, inferring the foreground object is basketball given a crop showing a basketball player. Ideally, the misrepresented set should be empty but contains a small portion of examples due to chance. *Déjà vu* memorization occurs for memorized samples where the embedding of SSL_B does not encode the label but the embedding of SSL_A does. To measure the pervasiveness of *déjà vu* memorization, we compare the size of the memorized and misrepresented sets. Figure 5 shows how the four categories of examples change with number of training epochs and training set size. The unassociated set is not shown since the total share adds up to one. The misrepresented set remains under 5% and roughly unchanged across all settings, consistent with our explanation that it is due to chance. In comparison, VICReg's memorized set surpasses 15% at 1000 epochs. Considering that up to 5% of these memorized examples could also be due to chance, we conclude that **at least 10% of VICReg's training set is *déjà vu* memorized.**

4.5. Visualizing *Déjà Vu* Memorization

Beyond enabling label inference using a periphery crop, we show that *déjà vu* memorization allows the SSL model to encode other forms of information about a training image. Namely, we train an RCDM [29] on the public dataset $\mathcal{X}$ and use it to visually reconstruct training images given their periphery crop. We aim to answer the following two questions: **(1)** Can we visualize the distinction between correlation and *déjà vu* memorization? **(2)** What foreground object details can be extracted from the SSL model beyond class label?



(a) A correlated dam example **(b)** A memorized dam example

Fig. 6. Correlated and Memorized examples from the *dam* class. Both SSL_A and SSL_B are SimCLR models. **Left:** The periphery crop (pink square) contains a concrete structure that is often present in images of dams. Consequently, the trained RCDM can reconstruct the foreground object using representations from both SSL_A and SSL_B through this correlation. **Right:** The periphery crop only contains a patch of water. The embedding produced by SSL_B only contains enough information to infer that the foreground object is related to water, as reflected by its KNN set and RCDM reconstruction. In contrast, the embedding produced by SSL_A memorizes the association of this patch of water with dam and the RCDM can visualize the embedding to produce images of dams.

Reconstruction pipeline. RCDM is a conditional generative model that is trained on the *backbone embedding* of images $X_i \in \mathcal{X}$ to generate an image that resembles X_i. All training images are first face-blurred for privacy purposes. Bordes et al. [29] showed that the backbone embedding of SSL models contains more low-level information about the image, making them better suited for conditioning the RCDM. At test time, following the pipeline in Figure 2, we first use the projector embedding to find the KNN subset for the periphery crop, $\text{crop}(A_i)$, and then average their backbone embeddings as input to the RCDM model. Ideally, when the public set contains enough representative images, the average representation of the KNN subset encodes objects present in A_i, and the RCDM model decodes this representation to visualize these objects.

Visualizing Correlation vs. Memorization. Figure 6 shows examples of dams from the correlated set (left) and the memorized set (right) as defined in Section 4.4.2, along with the associated KNN set and RCDM reconstruction. Both SSL_A and SSL_B are SimCLR models. In Figure 6a, the periphery crop is represented by the pink square, which contains concrete structure attached to the dam's main structure. As a result, both SSL_A and SSL_B produce embeddings of $\text{crop}(A_i)$ whose KNN set in $\mathcal{X}$ consist of dams, *i.e.*, there is a correlation between the concrete structure in $\text{crop}(A_i)$ and the foreground dam. The RCDM reconstructions also consist of dams or structures that closely resemble dams. In Figure 6b, the periphery crop only contains a patch of water, which does not strongly correlate with

(a) Memorized European badgers **(b)** Memorized American badgers

Fig. 7. Visualization of *déjà vu* memorization beyond class label. Both SSL_A and SSL_B are VICReg models. The four images shown belong to the memorized set of SSL_A from the *badger* class. RCDM reconstruction using embeddings from SSL_A can reveal not only the correct class label, but also the specific badger species: *European* (left) and *American* (right). Such information does not appear to be memorized by the reference model SSL_B.

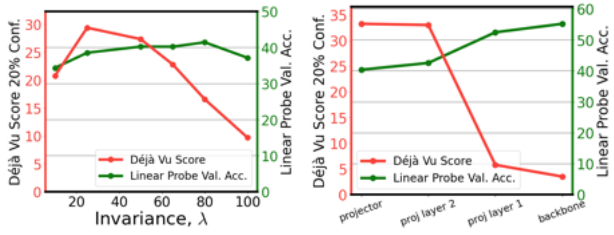
dams in the ImageNet distribution. Evidently, the reference model SSL_B embeds $\text{crop}(A_i)$ close to that of other objects commonly found in water, such as sea turtle and submarine. In contrast, the KNN set according to SSL_A all contain dams despite the vast number of alternative possibilities within the ImageNet classes, and the RCDM reconstruction outputs dams as well which highlight memorization in SSL_A between this specific patch of water and the dam.

Visualizing Memorization Beyond Class Label. We now use our reconstruction algorithm to show that *déjà vu* memorization can be exploited to reveal detailed information beyond class label. Figure 7 shows four examples of badgers from the memorized set. In all four images, the periphery crop (pink square) does not contain any indication that the foreground object is a badger. Despite this, the KNN set and the RCDM reconstruction using SSL_A consistently produce images of badgers, while the same does not hold for SSL_B. More interestingly, reconstructions using SSL_A in Figure 7a all contain *European* badgers, while reconstructions in Figure 7b all contain *American* badgers, accurately reflecting the species of badger present in the respective training images. Since ImageNet-1K does *not* differentiate between these two species of badgers, our reconstructions show that SSL models can memorize information that is highly specific to a training sample beyond its class label[8].

4.6. Mitigation of *déjà vu* memorization

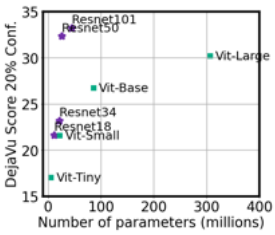
We cannot yet make claims on why *déjà vu* occurs so strongly for some SSL training settings and not for others. To gain some intuition for future work, we present additional observations that shed light on which parameters have the most salient impact on *déjà vu*

[8]See Appendix C.4 for additional visualization experiments.



(a) Loss hyperparameter **(b)** Guillotine regularization

Fig. 8. Effect of two kinds of hyperparameters on VICReg memorization. **Left:** *déjà vu* score (red) versus the *invariance* loss parameter, λ, used in the VICReg criterion (100k dataset). Larger λ significantly reduces *déjà vu* , with minimal effect on linear probe validation performance (green). $\lambda = 25$ (near maximum *déjà vu*) is recommended in the original paper **Right:** *déjà vu* score versus projector layer—guillotine regularization [30]—from projector to backbone. Removing the projector can significantly reduce *déjà vu* . Appendix C.3.6 shows that the backbone still can memorize, however; we demonstrate reconstructions using the SimCLR backbone.



(a) *déjà vu* vs. capacity

Criteria	DV	Acc P/B
Supervised	8.9	55.3/61.1
Byol[93]	8.0	54.3/59.4
SimCLR[49]	10.0	44.2/54.1
Dino[42]	14.5	26.3/55.7
Barlow T.[236]	30.5	33.7/54.4
VICReg[23]	**33.2**	40.3/55.2

(b) *déjà vu* (DV) vs. Criterion

Fig. 9. Effect of model architecture and criterion on *déjà vu* memorization. **Left:** *déjà vu* score with VICReg for resnet (purple) and vision transformer (green) architectures versus number of model parameters. As expected, memorization grows with larger model capacity. This trend is more pronounced for convolutional (resnet) than transformer (ViT) architectures. **Right:** Comparison of *déjà vu* score 20% conf. and ImageNet linear probe validation accuracy (P: using projector embeddings, B: using backbone embeddings) for various SSL criteria.

memorization.

Déjà vu memorization worsens by increasing number of training epochs. Figure 4a shows how *déjà vu* memorization changes with number of training epochs for VICReg. The training set size is fixed to 300K samples. From 250 to 1000 epochs, the *déjà vu* score (red curve) grows *threefold*: from under 10% to over 30%. Remarkably, this trend in memorization is *not* reflected by the linear probe gap (dark blue), which only changes by a few percent beyond 250 epochs.

Training set size has minimal effect on *déjà vu* memorization. Figure 4b shows how *déjà vu* memorization responds to the model's training set size. The number of training epochs is fixed to 1000. Interestingly, training set size appears to have almost *no* influence on the *déjà vu* score (red line), indicating that memorization is equally prevalent with a 100K dataset and a 500K dataset. This result suggests that *déjà vu* memorization may be detectable even for large datasets. Meanwhile, the standard linear probe train-test accuracy gap *declines* by more than half as the dataset size grows, failing to represent the

memorization quantified by our test.

Training loss hyper-parameter has a strong effect. Loss hyper-parameters, like VICReg's invariance coefficient (Figure 8a) or SimCLR's temperature parameter (Appendix Figure 2a) significantly impact *déjà vu* with minimal impact on the linear probe validation accuracy.

Some SSL criteria promote stronger *déjà vu* memorization. Table 9b demonstrates that the degree of memorization varies widely for different training criteria. VICReg and Barlow Twins have the highest *déjà vu* scores while SimCLR and Byol have the lowest. With the exception of Byol, all SSL models have more *déjà vu* memorization than the supervised model. Interestingly, different criteria can lead to similar linear probe validation accuracy and very different degrees of *déjà vu* as seen with SimCLR and Barlow Twins. Note that low degrees of *déjà vu* can still risk training image reconstruction, as exemplified by the SimCLR reconstructions in Figures 6 and 9.

Larger models have increased *déjà vu* memorization. Figure 9a validates the common intuition that lower capacity architectures (Resnet18/34) result in less memorization than their high capacity counterparts (Resnet50/101). We see the same trend for vision transformers as well.

Guillotine regularization can help reduce *déjà vu* memorization. Previous experiments were done using the projector embedding. In Figure 8b, we present how Guillotine regularization[30] (removing final layers in a trained SSL model) impacts *déjà vu* with VICReg[9]. Using the backbone embedding instead of the projector embedding seems to be the most straightforward way to mitigate *déjà vu* memorization. However, as demonstrated in Appendix C.4.1, backbone representation with low *déjà vu* score can still be leveraged to reconstruct some of the training images.

4.7. Conclusion

We defined and analyzed *déjà vu* memorization, a notion of unintended memorization of partial information in image data. As shown in Sections 4.4 and 4.5, SSL models can largely exhibit *déjà vu* memorization on their training data, and this memorization signal can be extracted to infer or visualize image-specific information. Since SSL models are becoming increasingly widespread as foundation models for image data, negative consequences of *déjà*

[9]Further experiments are available in Appendix C.3.6.

vu memorization can have profound downstream impact and thus deserves further attention. Future work should focus on understanding how *déjà vu* emerges in the training of SSL models and why methods like Byol are much more robust to *déjà vu* than VICReg and Barlow Twins. In addition, trying to characterize which data points are the most at risk of *déjà vu* could be crucial to get a better understanding on this phenomenon.

Prologue to Article 4

PUG: Photorealistic and Semantically Controllable Synthetic Data for Representation Learning, *Florian Bordes, Shashank Shekhar, Mark Ibrahim, Diane Bouchacourt, Pascal Vincent, Ari S. Morcos*, Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS 2023) Datasets and Benchmarks track.

Despite their impressive performances across many benchmarks, deep neural networks struggle to understand the world the same way we do. A photograph of a cow in a winter background will be a very rare occurrence. However, a human will easily recognize the cow despite the unusual background. Since neural networks are learning by association, they might learn to associate the concept of grass with the concept of cow which makes them unable to detect a cow on unusual backgrounds. Unfortunately, most benchmarks and evaluations lack fine-grained labeling, which would allow practitioners to detect if the network is relying on a spurious feature for classification. Since the cost of fine-grained labeling of real data is too high and uncertain, I decided to leverage a powerful video game engine (the Unreal Engine) for dataset generation. Most synthetic datasets available when writing this thesis were too toyish and lacked the realism needed to evaluate models trained on real images. With the Unreal Engine, we can significantly improve realism while keeping complete control over the environments. In this article, we introduced our setup called PUG (Photorealistic Unreal Graphics) and four datasets for Out-Of-Distribution (OOD) generalization, ImageNet benchmarking, vision-language model evaluation, and fine-tuning.

Contribution statement I started preliminary exploration for this project in 2021, and in 2022 Ari joined the project and provided very insightful guidance and support. I started developing an interactive web demo in which a user could change the factors of variation in a scene (like camera position, object orientation, textures, and backgrounds). When changing a factor, the images were passed through different neural networks, and their predictions were printed live on the screen. This demo allowed us to better grasp the failure modes of neural networks and to get people interested in the project. Afterwards, with Ari, we decided to buy a significant number of 3D assets, to have a diverse enough set of environments, to create

datasets that people could use to evaluate their models. I created the first *PUG: Animal* dataset, which was targeted towards OOD generalization by providing all combinations of a set of factors of variation. Then, I created a second dataset coined *PUG: ImageNet* as an additional benchmark for pre-trained ImageNet models. The third dataset I created is *PUG: SPAR* (Spatial, Position, Attribute, And Relation), which benchmark vision-language models (VLMs). Shashank joined the project later to create a fourth dataset called PUG: AR4T to show that in addition to using synthetic data for evaluation, we can also use them for fine-tuning CLIP-based models to improve reasoning abilities. Diane added the equivariance experiments while Mark ran the evaluations on PUG: ImageNet. Writing the paper was a common effort between all the co-authors. I made all the code on GitHub and developed the website.

Chapter 5

Article 4: PUG: Photorealistic and Semantically Controllable Synthetic Data for Representation Learning

Synthetic image datasets offer unmatched advantages for designing and evaluating deep neural networks: they make it possible to (i) render as many data samples as needed, (ii) precisely control each scene and yield granular ground truth labels (and captions), (iii) precisely control distribution shifts between training and testing to isolate variables of interest for sound experimentation. Despite such promise, the use of synthetic image data is still limited – and often played down – mainly due to their lack of realism. Most works therefore rely on datasets of real images, which have often been scraped from public images on the internet, and may have issues with regards to privacy, bias, and copyright, while offering little control over how objects precisely appear. In this work, we present a path to democratize the use of *photorealistic* synthetic data: we develop a new generation of interactive environments for representation learning research, that offer both *controllability* and *realism*. We use the Unreal Engine, a powerful game engine well known in the entertainment industry, to produce **PUG (Photorealistic Unreal Graphics)** environments and datasets for representation learning. In this paper, we demonstrate the potential of PUG to enable more rigorous evaluations of vision models. The datasets can be downloaded at `https://pug.metademolab.com/`.

5.1. Introduction

A grand goal of machine learning is to learn representations of data that are useful across many tasks. Essential to measuring and making progress towards this goal is the availability of ample controllable, realistic data for evaluation and training. This is especially true when considering deep neural network models not only in terms of their raw accuracy, but also their robustness and fairness—crucial properties for models deployed in real-world applications. However, collecting such data is challenging, presenting issues with privacy,

bias, and copyright. Furthermore, the majority of available image datasets lack fine-grained labels and are challenging to manipulate beyond coarse image augmentations (e.g. with a photograph, it is hard to change the viewpoint or the time of day).

Using synthetic image data where we precisely control all the factors affecting the rendered scene gives easy access to the corresponding rich set of factor labels. This enables evaluating the extent of a trained deep neural network's abilities, most importantly its robustness. Is the network robust to change in pose? Are the predictions similar for different textures? All these questions may be answered systematically by using synthetic data, enabling highly rigorous evaluations of deep neural network models. In addition, training could also benefit from controllable factors[1], by increasing the robustness of models with respect to these factors. They may also be used to monitor training, e.g. tracking which factors a model focuses on or becomes most invariant to, and in which order, as training progresses. This potentially enables better understanding of the training and generalization dynamics in deep neural networks. However the lack of realism typical in many of the currently available synthetic image datasets, and their usually very limited scope greatly limits their usefulness for general image representation learning research.

To address this, we introduce[2] a new family of synthetic *Photorealistic Unreal Graphics* (PUG) *datasets*, designed for ease of use by the representation learning research community, where image realism is significantly improved compared to current public synthetic image datasets. The environments were built using the Unreal Engine [72], which is widely used in the video game and entertainment industries and praised for its realism. In addition to pre-rendered static image datasets, we also introduce the TorchMultiverse python library, which offers a simple python interface to enable easily controlled dataset creation from any given PUG environment. Using these tools, we contribute 4 new datasets and show their usefulness across several different research domains. To summarize:

- We introduce a new family of environments and image datasets (coined as PUG) for representation learning, based on the Unreal Engine [72].
- We present *PUG: Animals* for research on out-of-distribution (OOD) generalization and to study the representational space of foundation models.

[1]We define factors here as distinctive attributes that describe the data, such as color or pose of an object.
[2]As a reminder, any use of content or technologies made available by Unreal and/or Epic Games, or any other provider, should comply with their applicable terms (such as the Content License Agreement available at `https://www.unrealengine.com/en-US/eula/content` or any other direct agreement you may have with Epic / Unreal)

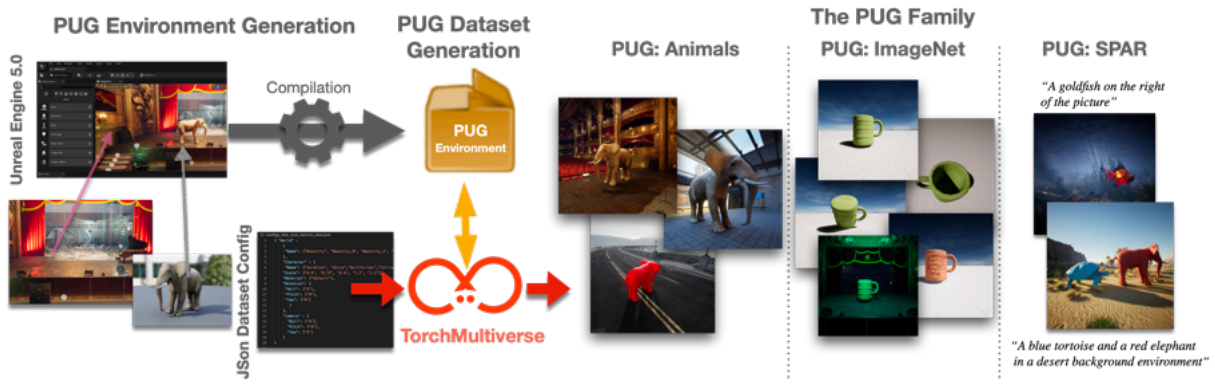


Fig. 1. The PUG Dataset Family (left) Cartoon illustration of our dataset creation setup, which consists of two steps: environment creation and then data creation. (right) Example images from PUG: Animals, PUG: Image-Net, and PUG: SPAR.

- We introduce *PUG: ImageNet* as an additional robustness test set to ImageNet, containing a rich set of factor changes such as pose, background, size, texture, and lighting.
- We introduce *PUG: SPAR* for evaluating vision-language models. We use it to demonstrate how synthetic data can be utilized to address known benchmark limitations. In addition, we introduce *PUG: AR4T* for fine-tuning vision-language models and use it to demonstrate the reliability of PUG: SPAR in contrast to other benchmarks.

5.2. Related work

Synthetic data for representation learning To address robustness shortcomings, researchers today commonly study representations using lower-fidelity controlled datasets such as CLEVR, Biased Cars, and ShapeNet [118, 147, 48]. Other datasets also contain precise factor labels useful for probing how well a representation encodes each factor in a structured form [87, 195, 221]. While these datasets offer control in terms of the factors that change as well as the train and evaluation splits enabling controlled scientific experimentation, they lack realism. This gap between the lower-fidelity controlled data and the real world poses a challenge for the broader application of these studies. On the other hand, photorealistic datasets have been explored in various application-specific domains in machine learning (outside of representation learning.) This is especially relevant when trying to evaluate and train models on rare events in which getting real data might be really difficult, such as for autonomous driving. CARLA [65] is a popular self-driving car simulator which offer highly realistic environment with a significant amount of controllable factors such as environmental conditions, full control of all static and dynamic actors and maps rendering. Another domain where simulated environments are commonly used is reinforcement learning (RL), as RL algorithms often requires the ability to run millions of simulations to learn to master

non-trivial tasks, and this cannot be done in a real environment. Data environments based on video games like Atari have been very popular to design and evaluate RL algorithms. Alternatively, platforms like Habitat [196] offers indoor scene for training home assistant agents. While these simulators, games or datasets can offer some photo-realism and mimic real world interactions for agents, they are relegated to domain-specific applications making them challenging to use for evaluating the representations of deep neural networks more broadly. Since our focus is not RL, we do not need to embed a fast simulator capable of rendering several thousands frames per second for effective online-training. Instead we can pre-render custom high-quality datasets offline. Photorealistic environments and datasets have also been explored for more general domains with the ThreeDWorld platform [80]. Based on the Unity game engine, it offers an interactive environment that can be leveraged to create datasets. The environment is presented as a simulator that is generic enough to handle multiple uses cases, and users can customize the setup of a scene and the data smapling through a low level API. One such dataset that utilizes ThreeDWorld is the Synthetic Visual Concepts (SyVIC) dataset [47], which uses the API to create scene images and descriptive captions for training vision-language models. One of the downsides of ThreeDWorld is that the back-end, the simulator itself, is closed source which limits external contributions. In contrast with ThreeDWorld, we do not provide a platform or a generic simulator for people to use. In fact, we believe that tools like the Unreal Engine are simple enough to be used directly by researchers to create the environments they want without the need to use an intermediate platform. In addition, being free of such intermediate platform allows us to leverage most of the content created for video gaming directly into our simulator by using the existing Epic Games marketplace.

Evaluating model robustness To study model robustness, there is an inherent trade-off between photo-realism and control. Photo-realism depicts objects as they appear in the real world, but often lacks control to precisely define the factors to describe the object such as pose or background. Prior works either collect natural images with specific factor changes [225, 20] or label distinctive factors in existing datasets [111]. Such datasets allow researchers to measure average accuracy on photo-realistic images but lack granular control necessary for precisely controlled robustness experiments. On the other hand, prior studies [110, 1, 3] examine model robustness with respect to factors such as pose and size by rendering 3D-objects such as buses. These studies precisely control how each object is depicted, but lack realism. In this work, we advance the photo-realism of these prior works by using the Unreal engine 5.0 [72], a rendering engine commonly used in high-end cinematic CGI and high-resolution video games which allows us to measure robustness with respect to factors of variation such as lighting.

Benefits and limitations of using generative models as data generator Another way to generate realistic datasets is to use generative models[106, 91]. However, one limitation of such models, despite impressive improvements in the last few years [60], is the lack of quality control on what the model can produce [81]. It's not uncommon to find cases in which the model will ignore parts of the conditioning prompt. Despite such limitations, many works have tried to leverage generative model as an additional source of data to train deep neural networks with some success [12, 19, 200, 13, 244, 138, 114, 115, 181, 100]. Another limitation of using generative models is privacy concerns that arise from such models replicating data from their training datasets [192]. Finally, Shumailov et al. [187] recently demonstrated that training on data recursively generated from such models results in increasing underestimates of the tails and overestimates of the mode, amplifying bias in datasets. In contrast to generative models that might produce unreliable results, we use an entirely controllable environment for which we can have a known and accurate generation with respect to a set of factors.

5.3. Photorealistic Unreal Graphics (PUG) environments and datasets

5.3.1. Leveraging Unreal Engine to create environments and datasets for representation learning

We introduce the *Photorealistic Unreal Graphics* (PUG) environments, a family of 3D graphics environments that leverage Unreal Engine for rendering image data for representation learning research. To create a PUG environment, we first obtain a number of assets [3] which can be 3D objects or 3D backgrounds. Then, we import them in the Unreal Engine editor and create blueprints that yield a simple generic 3D environment. Once this generic and controllable environment is created, it is compiled into a Linux binary file, which can be run on standard GPU clusters. This environment is programmed in such a way that when running, it is listening for incoming packets through WebRTC which can specify instructions about how to change a scene. Since most machine learning practitioners are used to python scripting, we wanted to have a very simple approach by which a user can request image data rendered from a packaged PUG environment, through very simple python code and JSON config files. To do so, we developed a python API, *TorchMultiverse*, that allows a user to easily specify a scene configuration in JSON and request rendered images from the PUG by using WebRTC. Once the factors have been set as requested by the user, for a specific environment configuration, the user can send a command to freeze the current environment and receive back an image. It takes around 1 second to render an image at a resolution of

[3]We purchased assets from the Epic Game Store and used assets from Sketchfab [55]. The complete list of assets we have used is available at `https://github.com/facebookresearch/PUG`

512x512 on a V100 GPU[4]. We illustrate this setup in Figure 1. It shows how starting from 3D assets, we design interactive environments that enable us to create different datasets. In the present work, we focus on pre-rendered static image datasets, however, our setup also allows dynamic communication between a PUG environment and a pytorch program, meaning that new data could be requested and rendered on the fly while training a deep neural network. We leave the exploration of such active learning setups, as well as the rendering of videos, as future work.[5]

5.3.2. PUG: Animals

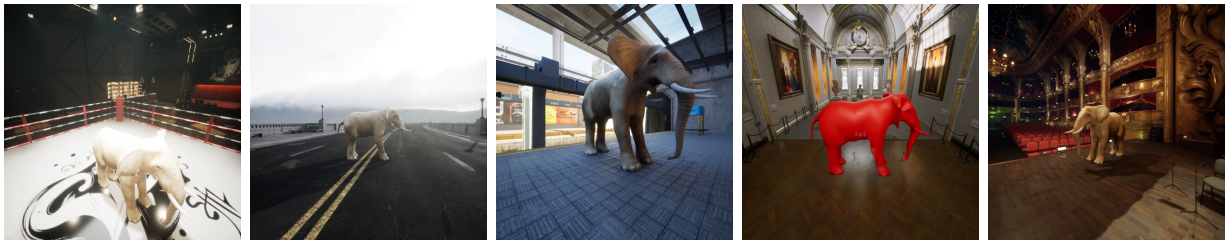


Fig. 2. We present ***PUG: Animals***, a new photorealistic synthetic dataset with annotated factors of variations to evaluate the out-of-distribution (OOD) robustness of models.

As the first member of the PUG family, we introduce *PUG: Animals* (Figure 2), which contains 215 040 pre-rendered images using 70 animals assets, 64 backgrounds, 3 object sizes, 4 textures, under 4 different camera orientations. PUG: Animals is designed with the intent to create a dataset with every combination of the factors of variation available. PUG: Animals allows one to precisely control distribution shifts between training and testing which can give researchers better insight on how a deep neural network generalizes on held out factors of variations. Surprisingly, the usage of 3D realistic synthetic data is limited in OOD generalization research – with the exception of Biased-cars[148] that has been used to study generalization on new category-viewpoints. Commons OOD datasets are Colored MNIST [7] – to study how well a network can generalize to unseen combinations of digits and colors and MNIST-R [83] – to study generalization on a new combination of digits and rotations. However, MNIST-based dataset might be too toyish to evaluate modern architectures. A more *realistic* dataset based on real images is Nico++[228] – to study generalization with respect to different domains or environments. However, in Nico++ the objects and backgrounds are never entirely disentangled (the context background is different for each image). Thus, it is never clear if the model is failing because of the context or

[4]In our setup, we paralyze the rendering across 64 GPUs. A dataset like PUG: Animals which contains 200K images has taken around 1h to be entirely rendered.

[5]It might also conceivably be used as a photorealistic interactive environment for reinforcement learning (RL), but the high quality image rendering achieved in this system currently appears too compute-intensive and slow to be practically useful in the context of current RL research. Our initial targeted research community and use case is that of supervised and self/unsupervised representation learning from image data, rather than RL.

because of a specific object (since the contexts and the objects are never disentangled).

In contrast, in PUG: Animals the animal asset is always the same, in that case, the environment factor and the objects are perfectly disentangled such that if the model is able to classify correctly an elephant on a road and is not able to classify the elephant in a museum, we can rigorously say that the failure is caused by the change in context background. In addition to analysis of the robustness with respect to the background scene, it is also possible to analyze with PUG: Animal the robustness with respect to the camera position, asset size, and texture (as we demonstrated in Appendix D.3).

Studying foundation model representational space PUG: Animal can also be to study the equivariance of foundation models' representations. For this, we augment each image in PUG: Animals with a caption that describes it according to its factor values (sizes are converted to three adjectives: "small", "medium" or "big", see Appendix D.3.1 for details), using the following template[6]: *"A photo of a [size] sized [character] textured with [texture] on a [background] background"*. Informally, equivariance of a model's representation with respect to a factor means that when the factor changes from one value to another, the embedding of the corresponding image (or caption) changes in a predictable manner. Equivariance is a sought-after property to improve sample efficiency and robustness to transformations [124, 197, 217]. Similar to previous works on equivariance and compositionality Bouchacourt et al. [34], Xie et al. [226], we measure equivariance as the alignment (i.e. parallelism) between embedding differences. First, we feed images and their corresponding captions to 9 pretrained vision-language models including multiple CLIP models Radford et al. [166], NegCLIP Yuksekgonul et al. [235] Flava Singh et al. [189], BLIP Li et al. [139] and X-VLM Zeng et al. [239] and collect their embeddings of PUG: Animals images and created captions. For each model, we compute difference vectors between the embeddings of two images (or captions) of an object undergoing a factor change: e.g. a big penguin textured with grass on a background "village square" modified to the same penguin but with background "restaurant", see arrows in Figure 3 (left). Specifically, for a sample i, undergoing a change from background b_k for background b_l, we denote the difference vector between the embedding of the image of the sample with backgorund b_k and the image of the same sample but with background b_l by $v^i_{b_k \to b_l}$. Similarly, we denote by $u^i_{b_k \to b_l}$ the difference vector between the embedding of each of the two captions accompanying the images.

Then, we measure the alignment of difference vectors across pairs *undergoing the*

[6]Note that the camera and character orientations are not described, as well as the texture when it is default.

same factor change (here, the penguin and the cat) as their cosine similarity[7]. We estimate three types of equivariance: (i) Image equivariance: how parallel (measured with cosine similarity) are difference vectors across image pairs? (lined and dashed red arrows) (ii) Text equivariance: same but for caption pairs (parallelism of lined and dashed green arrows) (iii) Across modalities equivariance: for the same object, alignment of difference vectors between pairs of image-caption (i.e. alignment of the two arrows for the penguin). Specifically, for image equivariance between sample i and j, for background change b_k to b_l, we compute:

$$sim(v^i_{b_k\to b_l}, v^j_{b_k\to b_l}) = \frac{{v^i_{b_k\to b_l}}^T v^j_{b_k\to b_l}}{||v^i_{b_k\to b_l}||\ ||v^j_{b_k\to b_l}||} \tag{5.3.1}$$

For text equivariance, we compute be $sim(u^i_{b_k\to b_l}, u^j_{b_k\to b_l})$ while for across equivariance, we compute $sim(v^i_{b_k\to b_l}, u^i_{b_k\to b_l})$. We report cosine similarity averaged over pairs and possible changes for each factor (higher value means higher equivariance, 1 is the maximum). Specifically, the image equivariance for background writes as

$$\frac{1}{B(B-1)}\sum_{b_k}\sum_{b_l}\frac{1}{N(N-1)}\sum_i\sum_j sim(v^i_{b_k\to b_l}, v^j_{b_k\to b_l}) \tag{5.3.2}$$

where B is the number of possible backgrounds and N is the number of samples.

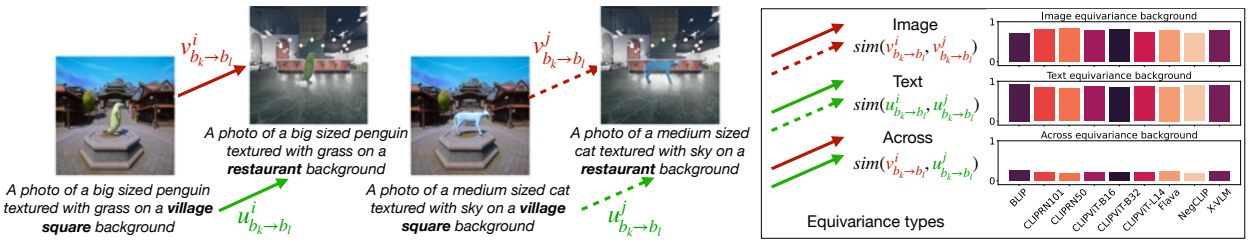


Fig. 3. Measuring foundation models equivariance thanks to PUG: Animals. Left: Illustration of how to use PUG: Animals to compute equivariance. **Right:** Image and text equivariance is present with respect to background, while across modalities equivariance to background doesn't hold as much. See main text for detailed results.

We show in Figure 3 (right) results for equivariance with respect to background. Plots for equivariance to texture and size are in Figure 5. Looking at Figure 3 results (right side, top row), we see that the foundation models' image embeddings present high equivariance to background (0.78 ± 0.04 on average over models). There is also (see Figure 5a) small image equivariance to texture (0.15 ± 0.04), but almost no equivariance to size (0.06 ± 0.02). Text equivariance is high with respect to background (average of 0.87 ± 0.03), but is also strong for size and texture (0.71 ± 0.11 for size and 0.81 ± 0.03 for texture, see Figure 5b) suggesting that foundation models' caption embeddings can be manipulated with vector arithmetic,

[7]Note that foundation model representations belong to the hypersphere, yet measuring equivariance as parallelism relies on Euclidean geometry, we discuss this in Appendix D.3.1. Still, cosine similarity is a starting point to showcase how PUG: Animals can be used to study models' representations.

similar to word vectors behaviours Ethayarajh et al. [76]. This aligns with the recent work of [201] that show linear behavior of VLMs text embedding spaces. Across modalities, small equivariance is present with respect to background (0.22 ± 0.03 and Figure 3 right side, bottom row). However when size or texture change for a given object, its image and caption representations seem to move in non-aligned directions (0.07 ± 0.01 for texture and 0.04 ± 0.01 for size, see Figure 5c). While more syntactically complex captions and other equivariance metrics could be designed, our aim here is to provide an initial study to showcase how PUG: Animals can be easily used to study state-of-the-art models representations.

5.3.3. PUG: ImageNet

Fig. 4. We present ***PUG: ImageNet***, a new photorealistic synthetic dataset with annotated factors of variations as an additional test set for ImageNet pretrained models.

As a second member of the PUG family, we introduce *PUG: ImageNet* (Figure 4), which contains 88,328 pre-rendered images using 724 assets representing 151 ImageNet classes with 64 backgrounds, 7 sizes, 9 textures, 18 different camera orientation, 18 different character orientation and 7 light intensity. In contrast to PUG: Animals, PUG: ImageNet was created by varying only a single factor at a time (which explain the lower number of images than PUG: Animals despite using more factors). The main purpose of this dataset is to provide a novel useful benchmark, paralleling ImageNet, but for *fine-grained evaluation of the robustness* of image classifiers, along several factors of variation.

An extensive evaluation of the robustness of SOTA models Our PUG: ImageNet dataset offers both photo-realism and precise control over how each object is depicted from pose and size to environment and camera-angle. We also provide a collection of objects with mappings to classes in the popular ImageNet dataset, enabling researchers to probe the robustness of SoTA vision models without retraining. We assess a variety of model architectures across several pretraining datasets including ImageNet-1/-21k, LAION (400M and 2B), and JFT300M [125, 143, 66]. We observe in Table 1 that the models that perform the best on the ImageNet validation accuracy are not always the ones which offer the best robustness on PUG: ImageNet. For example, the pretrained ViT-B32 trained on ImageNet-21k is better on the ImageNet validation set compared to a Swin-B, but offers worse robustness across all factors. We confirm no statistically significant relationship exists between ImageNet

accuracy and robustness by computing Pearson's correlation coefficients (Appendix D.3.3). This result showcases how PUG: ImageNet can be added as an additional benchmark to evaluate vision models.

		PUG: ImageNet Top-1 Accuracy across Factors of Variation					
	ImageNet Val.	Camera (Yaw,Pitch,Roll)	Pose (Yaw,Pitch,Roll)	Size	Texture	Light	Background
ResNet50	81.5	(38.1, 33.1, 26.9)	(38.0, 23.6, 22.9)	35.7	27.0	13.6	29.5
ResNet101	82.3	(43.4, 35.9, 29.4)	(45.1, 26.7, 25.6)	39.7	31.1	14.1	32.8
ViTLarge	85.8	(52.2, 40.4, 37.1)	(52.4, 30.4, 28.4)	46.4	42.9	8.9	34.6
ViTBasePretrained21k	84.3	(37.5, 34.3, 31.7)	(38.0, 21.8, 20.5)	33.0	28.5	4.1	26.6
Swin	83.6	(56.0, 45.6, 41.8)	(56.9, 35.3, 34.2)	52.9	40.1	19.1	42.0
BiT (JFT300M)	80.3	(40.5, 32.3, 26.0)	(42.1, 23.6, 22.8)	37.3	23.4	6.3	20.5
DINOv2 (LVD-142M)	84.5	(45.6, 41.1, 37.4)	(47.5, 28.8, 28.5)	43.1	35.0	6.1	30.9
Flava (PMD 70M)	75.5	(31.7, 23.4, 17.6)	(30.8, 17.6, 15.4)	30.5	24.2	7.8	21.9
CLIPViTB32 (400M)	62.9	(41.7, 30.2, 22.1)	(41.6, 23.8, 20.9)	40.1	34.4	5.7	24.4
CLIPViTB32 (2B)	66.6	(44.0, 31.5, 24.1)	(43.8, 24.8, 21.8)	42.2	34.7	3.3	26.0
CLIPViTL14 (400M)	72.8	(52.3, 39.8, 35.7)	(51.8, 29.0, 26.4)	50.6	41.1	4.3	33.0

Table 1. Robustness measured by average top-1 accuracy across factors on PUG: ImageNet (We show on the second column the traditional ImageNet validation set accuracy for comparison). Pretraining dataset sizes are indicated in parenthesis with the default being ImageNet-1k. CLIP uses ViT-B32 or ViT-L14. Camera orientation and object pose indicate accuracy along (yaw, pitch, roll) axes.

5.3.4. PUG: SPAR for VLMs

As a third member of the PUG family, we introduce *PUG: SPAR (Scene, Position, Attribute and Relation)* for evaluating vision-language models (VLMs). In contrast to pure vision based models, VLMs should be able to predict the correct caption (from a given set of captions) that describe the content of a given image. Several benchmarks to evaluate VLM models already exist such as Winoground [198] or ARO [235]. However, recent works[198, 61] have highlighted an important limitation in these benchmarks: some image-caption pairs in Winoground might be even too difficult to solve for a human whereas ARO has been shown by Lin et al. [141] to be mostly solvable without even using the image information at all. Consider that for an image containing a horse eating grass, ARO will propose two captions: *"the horse eating the grass"* and *"the grass eating the horse"*. The model should predict the correct caption between these two. However, the second caption is impossible, so even without looking at the image, any model can be confident that the first caption is the correct one.

Another shortcoming of current benchmarks is that most of them probe only if the model is correctly able to understand the relations or the attributes between objects. However, it is not clear if the failures in finding the correct relations or attributes come from the model not understanding them or come from not understanding which objects are present in the scene. For example, to understand complex relations like *A photo of an elephant on the left and a camel on the right in a desert background*, the model should first be able to identify whether the background of the picture is an actual desert. Then, the model should identify whether there is an elephant on the picture. It should understand what is *an elephant on the left*. The model could be very effective at identifying individual elephants or camels, but it could unexpectedly fail when a camel and an elephant appear in the same picture. If the

model does not fail in recognizing the animals, then we can probe the model to evaluate the position of each of them. We built PUG: SPAR with the goal of having a progressive evaluation scheme in which we can easily determine exactly what are the failure modes of a given VLM. From basic scene understanding to complex relations and attributes, our dataset offers a simple yet effective way to get a better understanding of the capabilities and limitations of VLMs.

The dataset contains 43,560 images with the associated factors of variations: 10 backgrounds, 32 animals, 4 relations (left/right, bottom/top), and 4 animal texture attributes (blue/red, grass/stone). We have images containing either 1) only the background (for scene recognition) 2) only one animal at a different left/right or bottom/top position 3) two animals at different left/right or bottom/top positions. And for each of the scenes (either single or multiple animals), we vary the texture of the animals and the background to evaluate the robustness of the model. Our setup and experiments are presented in Figure 2 in which we display some images from the dataset with the corresponding captions used for evaluations. In our benchmark, we used 6 different types of captions to evaluate the following: 1) Scene Recognition (first row) 2) Single animal classification (second row) 3) Single animal position detection (third row) 4) Multiple animals classification (fourth row) 5) Multiple animal position detection (fifth row) 6) Multiple animal and textures prediction (sixth row). For each of them, we evaluate the top-1 retrieval accuracy of the correct captions within the set of captions associated with each setup. We evaluate multiple models on these setups: OpenAI CLIP [166], OpenCLIP [112], Flava [189], BLIP [139], X-VLM [239] and NegCLIP [235]. Most of the models are correctly able to solve the scene recognition task (which is not surprising since we used only 10 environments that are very different from each other). Concerning the simple object recognition task when using a single animal, the performances across models is highly variable. Our experiments also highlight that the VLM performance in a multiple animals detection setting are much worse than the performance in a single animal detection setting. Those experiments show that despite their successes, VLMs are far from having a good understanding of the world and that improving the robustness of these models is a needed step for real-world robustness.

Inspired by Winoground [198], we present an experimental setup in which we leverage hard-negative pair of images. Instead of performing caption retrieval within all captions associated to a given setup, we performed caption retrieval between the correct and the *hard negative* caption. For example, the hard negative caption of *"An elephant on the left of the picture and a camel on the right of the picture"* will be *"A camel on the left of the picture and an elephant on the right of the picture"*. In addition of switching the relation (left/right and bottom/top), we also provide hard negative captioning for the attributes (blue/red and

grass/stone). In Table 3, we present our results using the hard-negative pair. We clearly observe that none of the models are able to predict the correct captions, with many models being close to random performance (50%).

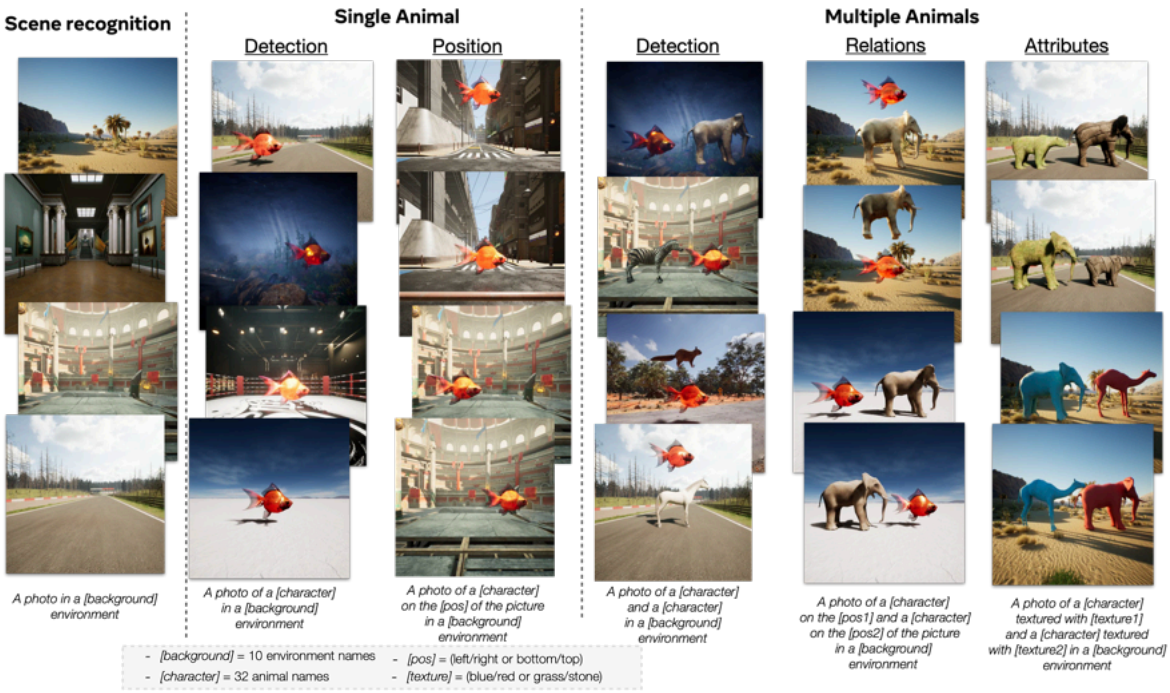
5.3.4.1. PUG: AR4T. Lastly, we introduce PUG: AR4T (Attributes and Relations for training). In contrast to PUG: SPAR which is only an evaluation benchmark, PUG: AR4T was created as an additional fine-tuning dataset for VLMs.[8] As shown in the previous section, VLMs struggle to understand spatial relations or attributes and thus are good candidates for our fine-tuning scenario. PUG: AR4T contains 249,986 training images with captions and 23,216 test images[9]. In Table 4, we present CLIP fine-tuning results on the ARO and PUG: SPAR benchmark. We also compare our results against Syn-CLIP, which is CLIP fine-tuned on the SyVIC synthetic dataset. Our results are very similar to Syn-CLIP, but Syn-CLIP training requires several additional tricks to arrive at this performance (Section 3.2 in [47]). On the other hand, the photo-realistic nature of PUG: AR4T enables us to match Syn-CLIP without any of these additional bells and whistles. However, even if we note some improvements on ARO, we are still far from having a model able to understand spatial relations. This is highlighted by the results given on our PUG: SPAR benchmark for which the improvement on single animal position prediction is still only above random chance while there is no improvement on the double animal location prediction task. This confirms the unreliability of the ARO benchmark highlighted by Lin et al. [141].

5.4. Conclusion

The fine-grained controllability of synthetic rendered image data makes it ideal for designing challenging evaluation benchmarks to better understand the properties and limitations of vision models, as well as for controlled training scenarios – if only it was closer to real data. To this effect, we introduced PUG datasets for representation learning. By leveraging the photorealism of the Unreal Engine[72], we created 4 new datasets and showcased their utility for robust evaluation. We showed how PUG: Animals could be leveraged for OOD generalization and to study properties of the representation spaces. We developed PUG: ImageNet as a new challenging benchmark that researchers can easily use to assess and compare the robustness of image classifiers. With PUG: SPAR we provide a reliable benchmark for vision-language models while PUG:AR4T offers additional data that could be leveraged to fine-tune VLMs. Together, the PUG family of datasets represents a new standard of photorealism and control for synthetic image data.

[8]The assets used to create PUG: SPAR and PUG: AR4T are different enough such that PUG: SPAR is still a good benchmark to evaluate models fine-tuned with PUG: AR4T.

[9]We also run experiments with a version of this dataset which contain 1M images but as shown in Table 4, adding more images does not increase performance on PUG: SPAR. We only release publicly the version of PUG:AR4T that contain 249,986 images.



Caption	Texture	ViT-B-32 (OpenAI CLIP)	ViT-B-32 (OpenClip 2B)	ViT-L-14 (OpenAI CLIP)	ViT-L-14 (OpenClip 2B)	ViT-H-14 (OpenCLIP 2B)	ViT-G-14 (OpenCLIP 2B)	Flava	BLIP	XVLM	NegCLIP
"A photo in a [background] environment"	N/A	100.00	100.00	100.00	90.00	100.00	**100.00**	60.00	100.00	90.00	100.00
A photo of a [character] in a [background] environment	Default	39.61	41.80	78.44	60.00	70.23	**78.36**	31.41	52.73	44.84	32.42
	Blue/Red	27.66	29.84	59.84	45.00	50.78	**63.75**	15.94	34.69	25.62	24.69
	Grass/Stone	23.13	27.19	54.53	40.00	45.00	**53.91**	13.44	32.81	23.44	22.97
"A photo of a [character] on the (left/right) of the picture in a [background] environment"	Default	19.84	20.00	39.38	30.31	35.94	**42.19**	14.69	27.03	22.66	16.56
"A photo of a [character] and a [character] in a [background] environment"	Default	9.42	11.88	44.08	25.36	34.98	**50.66**	6.30	22.29	13.50	7.32
	Blue/Red	3.54	6.03	23.00	14.16	14.59	**28.54**	1.93	7.02	2.49	2.94
	Grass/Stone	2.89	4.94	16.98	11.21	12.21	**21.89**	1.11	5.67	2.48	3.28
"A photo of a [character] on the left and a [character] on the right in a [background] environment"	Default	4.82	6.75	22.20	13.00	20.94	**29.83**	3.43	11.51	7.42	4.92
"A photo of a [character] textured with [texture1] and a [character] textured with [texture2] in a [background] environment"	Blue/Red	1.69	3.03	9.18	6.37	8.25	**15.89**	1.73	4.20	2.50	1.44
	Grass/Stone	1.20	2.24	8.46	6.14	7.08	**13.50**	1.50	3.50	1.89	1.52

Table 2. Setup and zero-shot evaluation of CLIP models on PUG: SPAR with **caption retrieval**. By using synthetic data, we can increase progressively the *difficulty* of a scene. Our setup is presented in the image above the table in which we show 6 different types of image captioning. 1) caption for background scene recognition for which we have 10 different backgrounds which are easy to distinguish from each other. 2) caption for single animal class prediction, the model should predict the correct categories over the 32 possible animals and 10 backgrounds (for a total of 320 captions). 3) caption for single animal position prediction that increases the number of caption up to 640 and lead to a significant drop in accuracy for every models. 4) caption for two animals class prediction, the model should predict the correct categories of the two animals presented in the images (5120 captions). 5) caption for two animals positions prediction, the model should predict the position of the two animals in the picture (over a total of 10240 captions). 6) caption for two textured animals class prediction, the model should recognize a blue elephant from a red camel. The performances of several VLMs models are presented in the table for which each row corresponds to one of the scenario described previously.

Caption	Values	B-32 CLIP	B-32 OpenClip 2B	L-14 CLIP	L-14 OpenClip 2B	H-14 OpenCLIP 2B	G-14 OpenCLIP 2B	Flava	BLIP	XVLM	NegCLIP
"A photo of a [character] on the [position] of the picture in a [background] environment"	Left/Right	49.53	47.66	50.00	46.72	49.84	50.31	49.22	51.88	**52.03**	48.91
	Bottom/Top	54.84	52.34	**66.87**	60.62	56.56	58.91	50.47	54.84	53.28	54.06
"A photo of a [character] on the [position1] and a [character] on the [position2] in a [background] environment"	Left/Right	53.88	55.62	53.17	**56.23**	55.74	54.44	54.41	53.79	55.02	54.36
	Bottom/Top	51.15	53.84	54.06	53.51	57.24	56.49	55.23	**60.26**	58.87	54.09
"A photo of a [character] textured with [texture1] and a [character] textured with [texture2] in a [background] environment"	Blue/Red	52.77	53.63	54.43	56.94	55.48	54.42	54.22	**57.32**	56.19	51.74
	Grass/Stone	52.79	54.14	56.31	57.28	56.62	**57.19**	54.53	53.94	54.26	49.92

Table 3. We present the performances of several VLMs with **hard negatives captioning** on PUG: SPAR in which we perform retrieval between two captions: the correct caption and the hard-negative corresponding caption. In that instance, the model should choose the correct caption between both of them (the probability to get the correct one with a random model would be 50%). Interestingly, none of the model presented in this table seem to be able to get a real understanding of simple position (left, right, bottom, top) or colors.

	ARO					PUG: SPAR (left/right)	
	VG-Relation (Macro-Accuracy%)	VG-Attribution (Macro-Accuracy%)	COCO-Order (Precision@1)	Flickr30k-Order (Precision@1)	Average	Single (Precision@1)	Double (Precision@1)
CLIP-ViT-B/32 (400M)	59.16 | 55.50	62.18 | 61.52	47.96	59.98	57.32	49.84	54.42
+ *FT w/ Syn-CLIP*	**71.40**	**66.94**	59.06	70.96	67.09 (+9.77)	N/A	N/A
+ *FT w/ PUG:AR4T (200K)*	68.36 | 75.18	65.54 | 64.44	57.80	69.74	65.36 (+8.04)	50.78(+0.94)	54.23(-0.19)
+ *FT w/ PUG:AR4T (1M)*	71.03 | 76.57	65.15 | 64.32	**61.07**	**72.84**	**67.52** (+10.3)	50.16(+0.32)	54.19(-0.23)

Table 4. Fine-tuning CLIP on PUG: AR4T. For VG-Relation and Attribution, the results (Acc1 | Acc2) indicate macro-accuracy across all relations and attributes (Acc1), and macro-accuracy on the subset of relations and attributes present in both ARO and PUG (Acc2). For PUG: SPAR, we evaluate on images in which there is only one animal (Single) or two animals (Double) with the relation being left or right. We were not able to run SynCLIP on PUG: SPAR because the model was not public at the time of the publication.

Chapter 6

Conclusion and Discussion

6.1. Summary of the contributions presented in this thesis

Precisely assessing the performances of deep neural networks is crucial to understand better what these systems learn and their limitations. With deep neural networks taking a more prominent place in our lives, we must create and design evaluation systems to ensure their safe deployment. Even if the road towards a complete and reliable set of benchmarks to evaluate the safety of AI systems is still long ahead, I introduced in this thesis key components that will help researchers and practitioners better understand the limitations of current neural networks.

- By providing a new **state-of-the-art qualitative evaluation method with RCDM**, researchers are now able to visualize and better understand which information is retained in a given representation – this without the need to gather extra labeling information. RCDM significantly impacted the SSL community by providing key visualizations for MSN[9] and I-JEPA[10]. In addition, it is a cited conditional generative model predecessor of DALL · E 2[167], which was a qualitative breakthrough in text-to-image generation.
- To better characterize the performance of neural networks, it is essential to perform **evaluation across different layers with Guillotine Regularization**. Otherwise, the results presented might be biased depending on the choice of training hyperparameters or tasks. This is especially important for SSL methods since the pretext training task is typically different from the downstream task.
- When evaluating neural networks, it is important to quantify how much they memorized their training data. To do that, I introduced the **Déjà vu Score**, a metric targeted at join-embedding Self-Supervised models that highlights how much they retain precise information about their training data.

- Lastly, I introduced **PUG, a new generation of datasets and benchmarks that enable photo-realistic and controllable generation of synthetic data**. In contrast to previous works, which offered toyish-looking objects and scenes, we leveraged video game engines and bought video game assets to build diverse and rich scene environments. We demonstrated how these datasets can be used for OOD research, equivariance studies, model robustness evaluation, vision-language model spatial relation understanding, and CLIP fine-tuning.

6.2. Other related contributions

In this thesis, I presented four articles. However, during my thesis I also co-authored other articles that could be of interest to the reader.

A cookbook of self-supervised learning [17] In this work, we offer a complete review of Self-Supervised Learning along with some guidance and tips on how to develop, improve, and evaluate SSL systems.

Towards Democratizing Joint-Embedding Self-Supervised Learning [32] In this work, I introduced a library called FFCV-SSL (based on the FFCV library of Leclerc et al. [129]). I developed this library to yield substantial speedup for training SSL methods up to 6 times faster, and demonstrated that one could train an SSL method like SimCLR (which had the reputation of being costly and long to train) in a few hours on an 8 GPU cluster or around a day in a single GPU setup. This work was a crucial enabler of several papers presented in this thesis, since it allowed us to speed up all our experiments significantly. The library is available at the following URL: `https://github.com/facebookresearch/FFCV-SSL`. It also gave us significant speed-ups for our evaluation pipelines.

A surprisingly simple technique to control the pretraining bias for better transfer: Expand or Narrow your representation [33] This work follows up on the findings from Bordes et al. [29] and Bordes et al. [30], and is a thorough empirical exploration of how changing the backbone dimension impacts the performances of Self-Supervised models. This research was primarily motivated by the observation that about all SSL models were based on the same architecture as their supervised counterparts. However, since SSL models extract more information about their inputs, the hypothesis I had was that wider representations might be beneficial for SSL training. One of the main findings was that supervised methods yield better in-distribution generalization if the size of the backbone representation is close to the number of classes. However, when using such a network for transfer learning to a different distribution, it is preferable to use a wider

backbone. Our finding is that the performance of SSL methods increases significantly across various benchmarks if we merely increase the size of their backbone representation. We are also the first to show that when using a linear projector to train SSL methods, the ratio between the projector and backbone dimensions is an important parameter to adjust.

The hidden uniform cluster prior in self-supervised learning [8] In this work, we show that many SSL methods are learning semantic information that is highly dependent on how the mini-batch is constructed. If the dataset is built in such a way that its classes are distributed uniformly, by taking a random mini-batch that is big enough, we can expect to get at least one image per class in the mini-batch. Then the most discriminative information that is used by SSL methods to learn to differentiate each image within a mini-batch is the class label. Because of that, the SSL model will be pushed to learn class related features since it will be the easiest way to solve the contrastive objective. However, if we do not have a uniform class distribution within a mini-batch, SSL performances can drop significantly. In this paper, we show that most SSL methods do not work well with unbalanced data, which breaks the promise of using SSL to leverage unlabelled and uncurated data. We advise researchers to evaluate their model on imbalanced data when introducing a new SSL method.

Objectives Matter: Understanding the Impact of Self-Supervised Objectives on Vision Transformer Representations [184] In this work, we analyze with several metrics how different SSL training objectives, ranging from join-embeddings SSL methods (like Dino) to reconstruction based SSL methods (like MAE), impact the representations learned by Vision transformers.

6.3. A path towards more principled evaluations

In this thesis, I presented building blocks to create better benchmarks for AI systems. The benchmarks should be adapted and improved as these AI systems progress and improve. In addition, we can expect that soon, AI systems will be regulated such that they will have to follow a set of specific standards before being deployed. Building such standards requires foreseeing the capabilities of the AI systems of tomorrow. In this section, I present some ideas that could help us to establish new standards for the evaluation of AI systems.

6.3.1. Towards a more complete evaluation suite for vision systems

With PUG: ImageNet, we introduced a dataset to evaluate pre-trained models across the following factors: backgrounds, camera orientation, object orientation, object texture, and scale, as well as scene lightning. To offer a complete benchmark, we would need more factors, such as:

- a variety of geographical locations that could match the diversity we have in the real world.
- weather variation: heavy rain, snow, or fog with different intensity levels.
- partial occlusion where objects can be behind fences, leaves, or windows.
- light variations: natural lighting that depend on the time of the day and artificial lightning.
- simulating different kinds of cameras.

Such a more complete evaluation system would enable the creation of precise model cards, specifying in which conditions a given model is expected to work and its limitations. We can expect that such model cards might become a requirement when open-sourcing models (or any AI-based applications) in the future. For example, when buying a pair of smart glasses equipped with an AI agent, we should know whether this agent will work well in outdoor environments, and if so, if it will also work well enough in rainy/foggy/snowy environments.

6.3.2. A dive into Vision Language Models

There has recently been a surge in interest in vision language models and their capabilities. Commons benchmarks like ARO[235] rely on finding the correct caption associated with given images. By doing that, we can measure if a given VLM can understand a relation such as "an object being on the left or the right of another object". However, those models often cheat by relying on natural statistical occurrences in the text. For example, a coffee cup will probably always be *on* a table and not *under* the table. Consequently, many relation benchmarks can be largely solved without using the information in the image. To evaluate if a model can understand what an image contains, we can leverage endless variations of scenes and captions with the PUG framework, including unusual ones, so that natural statistical occurrences will not bias the benchmarks. However, we should go beyond simple relations like left/right, top/bottom, that we used for PUG: SPAR. The next generation of benchmarks for vision language models should include rich labeled scenes, for which we will be able to ask precise questions such as: "What color is the dog's tail?", "How many books are in the library?", "What color is the 23rd book from the left located on the fifth shelf in the bookshelf that we can see on the right of the picture?"

6.3.3. Will neural networks be able to handle uncertainty?

An essential component of creating an autonomous intelligent machine system is the ability to manage uncertainty, by learning to predict the situations that are likely to happen given a specific context. A coffee mug is more likely to be next to a computer, whereas a watering can is more likely to be next to a plant. Consequently, when having only an image crop containing a computer and a handle, the model should predict that the handle is more

likely to be part of a coffee mug than a watering can. If we have videos, from a few frames, we should be able to predict several most likely future possibilities. A video dataset like Moving-MNIST[194] has either random or deterministic digit trajectories. When the digit is inside the video frame, its trajectory is deterministic. However, its trajectory is random when the digit bounces back from a side. A model that can handle uncertainty should be able to predict what is deterministic while displaying uncertainty on the factors with randomness in them. Thus, when the digit bounces back from a side, the model should predict several different probable future trajectories, while it should predict a single immediate trajectory continuation when the digit is inside the frame with no contact to the borders. Being able to properly handle and model this kind of uncertainty is a current missing but much-needed piece towards more reliable AI systems. But a huge gap exists between a simple dataset such as Moving-MNIST and real videos. Using PUG, we could create image and video datasets to control the number of predictable and random factors, while having more realistic scenes. If we look back at our smart AI glasses examples, there will be countless situations in which some objects might be occluded. In that instance, having a PUG dataset focused on occlusion could be an excellent way to evaluate how robust a model is to classification uncertainty in the presence of occlusions (partial or full).

6.3.4. How to evaluate neural networks that can plan ?

Once neural networks can handle uncertainty, the next logical step is to think about planning. Let's take an example about a pedestrian possibly crossing a street. There are two plans that an autonomous driving system can follow: stop if the pedestrian crosses the street or continue if the pedestrian does not cross the street. However, there might be more complex plans, such as designing a complete itinerary in which there could be unanticipated road works. Computer games can also be excellent environments to evaluate planning, since gaming often requires precise actions in a changing environment. However, how do we ensure an AI-based planning system is safe and robust? When playing a game, we do not want the system to find a way to cheat. When encountering road work, we do not want the system to shut down if there are too many orange traffic cones[1]. When designing an evaluation benchmark for a planning based-AI system, it will be crucial to manage and test the robustness of the model to ensure it can adapt quickly to unexpected events. One way to do that is to randomly introduce adversarial events to see if this system can be tricked into a problematic behavior. In a PUG-based simulator, an AI agent that could walk/run or drive toward a specific destination should be tested when there are: random objects on the road, animals or humans moving around, falling buildings, and road signs that tell the AI to drive

[1]Otherwise, the system will never work in Montreal!

into a lake. The primary motivation is to find ways to asses whether the AI agent might be easily deceived in a treacherous environment.

6.3.5. The curse of dimensionality for benchmarking

Evaluating and benchmarking AI agents is a crucial step to ensure their safe deployment. However, by increasing the number of interactions they can have with the world, we are also significantly increasing the number of ways in which they could derail into unexpected and potentially problematic behaviors. An AI agent specialized in playing a video game might be easy to evaluate since the number of actions it can take is restricted in the game. The same is true for smart glasses, which could run a local on-device agent system that can be restricted to give travel tips. However, the more the AI agent can richly interact with a complex world through images/video/text/sound and maybe touch, the more it will be difficult to correctly asses its robustness. For example creating reliable autonomous aircrafts is actually much easier than creating reliable autonomous self-driving cars precisely because the number of possible unexpected situations on the ground is much higher than in the sky. When conceiving a general AI agents performing a wide range of actions while evolving on the ground of a busy city, the number of potential failure cases is exponentially larger than when designing a highly specialized agent with a restricted number of actions evolving in a highly predictable factory environment. Even with all the best benchmarks and evaluation systems that can be designed, we should always keep in mind that by increasing the complexity and the reach of the AI system we are developing, we are also significantly increasing the complexity of making benchmarks to evaluate such systems, and thus making it much harder to guarantee their safety.

6.4. Conclusion

In this thesis, I developed building blocks toward better and more fine grained evaluation of deep neural networks. There is often an urge within the research community to increase the performance of deep neural networks, which might come at the cost of correctly assessing the robustness and the safety of the system we are deploying. In this thesis, I introduced new methods and approaches to evaluate AI systems, which are essential to better understand what the system has truly learned. However, as science progrsses and AI systems become more general agents, it will likely become more and more challenging to thoroughly assess the reliability of the AI system we are deploying. Consequently, it may be wise to restrict the use of AI systems to specific scenarios under which their robustness can be easily measured and guaranteed.

References

[1] Amro Abbas and Stéphane Deny. Progress and limitations of deep networks to recognize objects in unusual poses. *arXiv preprint arXiv:2207.08034*, 2022.

[2] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[3] Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4845–4854, 2019.

[4] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. Deep canonical correlation analysis. In *International conference on machine learning*, pages 1247–1255. PMLR, 2013.

[5] Srikar Appalaraju, Yi Zhu, Yusheng Xie, and István Fehérvári. Towards good practices in self-supervised representation learning. *arXiv preprint arXiv:2012.00868*, 2020.

[6] Srikar Appalaraju, Yi Zhu, Yusheng Xie, and István Fehérvári. Towards good practices in self-supervised representation learning. In *NeurIPS Self-Supervision Workshop*, 2020.

[7] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization, 2020.

[8] Mahmoud Assran, Randall Balestriero, Quentin Duval, Florian Bordes, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, and Nicolas Ballas. The hidden uniform cluster prior in self-supervised learning, 2022. URL `https://arxiv.org/abs/2210.07277`.

[9] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXI*, page 456–473, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19820-5. doi: 10.1007/978-3-031-19821-2_26. URL `https://doi.org/10.1007/978-3-031-19821-2_26`.

[10] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. *arXiv preprint arXiv:2301.08243*, 2023.

[11] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture, 2023.

[12] Pietro Astolfi, Arantxa Casanova, Jakob Verbeek, Pascal Vincent, Adriana Romero-Soriano, and Michal Drozdzal. Instance-conditioned gan data augmentation for representation learning, 2023.

[13] Shekoofeh Azizi, Simon Kornblith, Chitwan Saharia, Mohammad Norouzi, and David J. Fleet. Synthetic data from diffusion models improves imagenet classification, 2023.

[14] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019.

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL `http://arxiv.org/abs/1409.0473`.

[16] Robert J. N. Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. In *Neural Information Processing Systems*, 2021.

[17] Randall Balestriero, Mark Ibrahim, Vlad Sobal, Ari Morcos, Shashank Shekhar, Tom Goldstein, Florian Bordes, Adrien Bardes, Gregoire Mialon, Yuandong Tian, Avi Schwarzschild, Andrew Gordon Wilson, Jonas Geiping, Quentin Garrido, Pierre Fernandez, Amir Bar, Hamed Pirsiavash, Yann LeCun, and Micah Goldblum. A cookbook of self-supervised learning, 2023.

[18] Borja Balle, Giovanni Cherubin, and Jamie Hayes. Reconstructing training data with informed adversaries. *arXiv preprint arXiv:2201.04845*, 2022.

[19] Hritik Bansal and Aditya Grover. Leaving reality to imagination: Robust classification via generated datasets, 2023.

[20] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

[21] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[22] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *ICLR*, 2022.

[23] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

[24] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 17–36, Bellevue, Washington, USA, 02 Jul 2012. PMLR. URL https://proceedings.mlr.press/v27/bengio12a.html.

[25] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.

[26] Yoshua Bengio, Frédéric Bastien, Arnaud Bergeron, Nicolas Boulanger–Lewandowski, Thomas Breuel, Youssouf Chherawala, Moustapha Cisse, Myriam Côté, Dumitru Erhan, Jeremy Eustache, Xavier Glorot, Xavier Muller, Sylvain Pannetier Lebeuf, Razvan Pascanu, Salah Rifai, François Savard, and Guillaume Sicard. Deep learners benefit more from out-of-distribution examples. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 164–172, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/bengio11b.html.

[27] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013. URL http://arxiv.org/abs/1206.5538. cite arxiv:1206.5538.

[28] Florian Bordes, Sina Honari, and Pascal Vincent. Learning to generate samples from noise through infusion training. In *International Conference on Learning Representations (ICLR)*, 2017.

[29] Florian Bordes, Randall Balestriero, and Pascal Vincent. High fidelity visualization of what your self-supervised representation knows about. *CoRR*, abs/2112.09164, 2021. URL https://arxiv.org/abs/2112.09164.

[30] Florian Bordes, Randall Balestriero, Quentin Garrido, Adrien Bardes, and Pascal Vincent. Guillotine regularization: Improving deep networks generalization by removing their head, 2022. URL https://arxiv.org/abs/2206.13378.

[31] Florian Bordes, Randall Balestriero, and Pascal Vincent. High fidelity visualization of what your self-supervised representation knows about. *Transactions on Machine Learning Research*, 2022. URL https://openreview.net/forum?id=urfWb7VjmL.

[32] Florian Bordes, Randall Balestriero, and Pascal Vincent. Towards democratizing joint-embedding self-supervised learning, 2023. URL https://arxiv.org/abs/2303.01986.

[33] Florian Bordes, Samuel Lavoie, Randall Balestriero, Nicolas Ballas, and Pascal Vincent. A surprisingly simple technique to control the pretraining bias for better transfer: Expand or narrow your representation, 2023.

[34] Diane Bouchacourt, Mark Ibrahim, and Ari Morcos. Grounding inductive biases in natural images: invariance stems from variations in data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 19566–19579. Curran Associates, Inc., 2021. URL `https://proceedings.neurips.cc/paper_files/paper/2021/file/a2fe8c05877ec786290dd1450c3385cd-Paper.pdf`.

[35] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=B1xsqj09Fm`.

[36] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 267–284, 2019.

[37] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[38] Nicholas Carlini, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, and Eric Wallace. Extracting training data from diffusion models. *arXiv preprint arXiv:2301.13188*, 2023.

[39] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018.

[40] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/70feb62b69f16e0238f741fab228fec2-Paper.pdf`.

[41] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[42] Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, and Julien Mairal Piotr Bojanowski Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[43] Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, and Julien Mairal Piotr Bojanowski Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[44] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[45] Rich Caruana. Learning many related tasks at the same time with backpropagation. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. MIT Press, 1994. URL `https://proceedings.neurips.cc/paper/1994/file/0f840be9b8db4d3fbd5ba2ce59211f55-Paper.pdf`.

[46] Arantxa Casanova, Marlène Careil, Jakob Verbeek, Michal Drozdzal, and Adriana Romero. Instance-conditioned gan. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[47] Paola Cascante-Bonilla, Khaled Shehada, James Seale Smith, Sivan Doveh, Donghyun Kim, Rameswar Panda, Gül Varol, Aude Oliva, Vicente Ordonez, Rogerio Feris, et al. Going beyond nouns with vision & language models using synthetic data. *arXiv preprint arXiv:2303.17590*, 2023.

[48] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[49] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[50] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *NeurIPS*, 2020.

[51] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2020.

[52] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[53] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

[54] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[55] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. *arXiv preprint arXiv:2212.08051*, 2022.

[56] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[57] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/n19-1423. URL `https://doi.org/10.18653/v1/n19-1423`.

[59] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

[60] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat GANs on image synthesis. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=AAWuCvzaVt`.

[61] Anuj Diwan, Layne Berry, Eunsol Choi, David Harwath, and Kyle Mahowald. Why is winoground hard? investigating failures in visuolinguistic compositionality. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2236–2250, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL `https://aclanthology.org/2022.emnlp-main.143`.

[62] Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pages 1422–1430, 2015.

[63] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL `https://proceedings.neurips.cc/paper/2016/file/371bce7dc83817b7893bcdeed13799b5-Paper.pdf`.

[64] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

[65] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[66] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[67] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[68] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=BJO-BuT1g`.

[69] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Theoretical Computer Science*, 9(3-4):211–407, 2013.

[70] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography*, pages 265–284. Springer, 2006.

[71] Cian Eastwood and Christopher K. I. Williams. A framework for the quantitative evaluation of disentangled representations. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=By-7dz-AZ`.

[72] EpicGames. URL `https://www.unrealengine.com`. Unreal Engine is a copyright of Epic Games, Inc. and its affiliates (collectively, "Epic"). Any use of images, datasets, or other content made available by Epic, including without limitation through the Unreal Engine Marketplace or the Epic Games Launcher, in connection with your use of the PUG environments and datasets we've outlined in this paper and released publicly in connection hereto (the "PUG environments and datasets") or otherwise, is subject to the Epic Content License Agreement available at `https://www.unrealengine.com/en-US/eula/content` or other agreement between you and Epic.

[73] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[74] Linus Ericsson, Henry Gouk, and Timothy M Hospedales. How well do self-supervised models transfer? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5414–5423, 2021.

[75] Patrick Esser, Robin Rombach, and Bjorn Ommer. A disentangling invertible interpretation network for explaining latent representations. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition, pages 9223–9232, 2020.

[76] Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. Towards understanding linear word analogies. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3253–3262, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1315. URL https://aclanthology.org/P19-1315.

[77] Utku Evci, Vincent Dumoulin, Hugo Larochelle, and Michael C Mozer. Head2Toe: Utilizing intermediate representations for better transfer learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6009–6033. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/evci22a.html.

[78] Vitaly Feldman. Does learning require memorization? a short tale about a long tail. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 954–959, 2020.

[79] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An {End-to-End} case study of personalized warfarin dosing. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 17–32, 2014.

[80] C Gan, J Schwartz, S Alter, M Schrimpf, J Traer, J De Freitas, J Kubilius, A Bhandwaldar, N Haber, M Sano, et al. Threedworld: A platform for interactive multi-modal physical simulation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[81] Rohit Gandikota, Joanna Materzynska, Jaden Fiotto-Kaufman, and David Bau. Erasing concepts from diffusion models. *arXiv preprint arXiv:2303.07345*, 2023.

[82] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé Iii, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[83] Muhammad Ghifary, W. Kleijn, Mengjie Zhang, and David Balduzzi. Domain generalization for object recognition with multi-task autoencoders. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2551–2559, 2015. URL https://api.semanticscholar.org/CorpusID:12825123.

[84] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=S1v4N2l0-.

[85] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

[86] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL `https://proceedings.mlr.press/v15/glorot11a.html`.

[87] Muhammad Waleed Gondal, Manuel Wuthrich, Djordje Miladinovic, Francesco Locatello, Martin Breidt, Valentin Volchkov, Joel Akpo, Olivier Bachem, Bernhard Schölkopf, and Stefan Bauer. On the transfer of inductive bias from simulation to the real world: a new disentanglement dataset. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/file/d97d404b6119214e4a7018391195240a-Paper.pdf`.

[88] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[89] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL `http://arxiv.org/abs/1412.6572`.

[90] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[91] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[92] Priya Goyal, Quentin Duval, Jeremy Reizenstein, Matthew Leavitt, Min Xu, Benjamin Lefaudeux, Mannat Singh, Vinicius Reis, Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Ishan Misra. Vissl. `https://github.com/facebookresearch/vissl`, 2021.

[93] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

[94] Chuan Guo, Brian Karrer, Kamalika Chaudhuri, and Laurens van der Maaten. Bounding training data reconstruction in private (deep) learning. *arXiv preprint arXiv:2201.12383*, 2022.

[95] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and*

pattern recognition, pages 770–778, 2016.

[96] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[97] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738, 2020.

[98] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

[99] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[100] Ruifei He, Shuyang Sun, Xin Yu, Chuhui Xue, Wenqing Zhang, Philip Torr, Song Bai, and XIAOJUAN QI. IS SYNTHETIC DATA FROM GENERATIVE MODELS READY FOR IMAGE RECOGNITION? In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=nUmCcZ5RKF`.

[101] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019. doi: 10.1109/JSTARS.2019.2918242.

[102] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019.

[103] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *NeurIPS*, 2017.

[104] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

[105] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html`.

[106] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020.

[107] Harold Hotelling. Relations between two sets of variates. In *Breakthroughs in statistics*, pages 162–190. Springer, 1992.

[108] William W Hsieh. Nonlinear canonical correlation analysis by neural networks. *Neural Networks*, 13(10):1095–1105, 2000.

[109] Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

[110] Mark Ibrahim, Quentin Garrido, Ari Morcos, and Diane Bouchacourt. The robustness limits of sota vision models to natural variation. *arXiv preprint arXiv:2210.13604*, 2022.

[111] Badr Youbi Idrissi, Diane Bouchacourt, Randall Balestriero, Ivan Evtimov, Caner Hazirbas, Nicolas Ballas, Pascal Vincent, Michal Drozdzal, David Lopez-Paz, and Mark Ibrahim. Imagenet-x: Understanding model mistakes with factor of variation annotations. *arXiv preprint arXiv:2211.01866*, 2022.

[112] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL `https://doi.org/10.5281/zenodo.5143773`. If you use this software, please cite it as below.

[113] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/ioffe15.html`.

[114] Ali Jahanian, Xavier Puig, Yonglong Tian, and Phillip Isola. Generative models as a data source for multiview representation learning. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=qhAeZjs7dCL`.

[115] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=99RpBVpLiX`.

[116] Bargav Jayaraman and David Evans. Are attribute inference attacks just imputation? *arXiv preprint arXiv:2209.01292*, 2022.

[117] Li Jing, Pascal Vincent, Yann LeCun, and Yuandong Tian. Understanding dimensional collapse in contrastive self-supervised learning. In *ICLR*, 2022.

[118] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *CVPR*, 2017.

[119] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. In *CVPR*, 2020.

[120] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/d89a66c7c80a29b1bdbab0f2a1a94af8-Paper.pdf.

[121] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR 2014)*, 2014.

[122] Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/ddeebdeefdb7e7e7a697e1c3e3d8ef54-Paper.pdf.

[123] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations, 2022. URL https://arxiv.org/abs/2204.02937.

[124] David Klee, Ondrej Biza, Robert Platt, and Robin Walters. Image to sphere: Learning equivariant features for efficient pose prediction. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=_2bDpAtr7PI.

[125] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning, 2020.

[126] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[127] Alex Krizhevsky. Learning multiple layers of features from tiny images. pages 32–33, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[128] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Learning representations for automatic colorization. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 577–593. Springer, 2016.

[129] Guillaume Leclerc, Andrew Ilyas, Logan Engstrom, Sung Min Park, Hadi Salman, and Aleksander Madry. ffcv. https://github.com/libffcv/ffcv/, 2022. commit xxxxxxx.

[130] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[131] Yann LeCun, Sumit Chopra, Raia Hadsell, Marc'Aurelio Ranzato, and Fu-Jie Huang. A tutorial on energy-based learning. In G. Bakir, T. Hofman, B. Schölkopf, A. Smola, and B. Taskar, editors, *Predicting Structured Data*. MIT Press, 2006.

[132] Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, page 896, 2013.

[133] J.M. Lee. *Introduction to Topological Manifolds*. Graduate texts in mathematics. Springer, 2000. ISBN 9780387950266. URL `https://books.google.fr/books?id=5LqQgkS3--MC`.

[134] Kuang-Huei Lee, Anurag Arnab, Sergio Guadarrama, John Canny, and Ian Fischer. Compressive visual representations. In *NeurIPS*, 2021.

[135] Karel Lenc and Andrea Vedaldi. Understanding image representations by measuring their equivariance and equivalence. *International Journal of Computer Vision*, 2018. doi: 10.1007/s11263-018-1098-y.

[136] Chunyuan Li, Jianwei Yang, Pengchuan Zhang, Mei Gao, Bin Xiao, Xiyang Dai, Lu Yuan, and Jianfeng Gao. Efficient self-supervised vision transformers for representation learning. In *ICLR*, 2022.

[137] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[138] Daiqing Li, Huan Ling, Seung Wook Kim, Karsten Kreis, Sanja Fidler, and Antonio Torralba. Bigdatasetgan: Synthesizing imagenet with pixel-wise annotations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 21298–21308. IEEE, 2022. doi: 10.1109/CVPR52688.2022.02064. URL `https://doi.org/10.1109/CVPR52688.2022.02064`.

[139] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, 2022.

[140] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL `http://arxiv.org/abs/1405.0312`.

[141] Zhiqiu Lin, Xinyue Chen, Deepak Pathak, Pengchuan Zhang, and Deva Ramanan. Visualgptscore: Visio-linguistic reasoning with multimodal generative pre-training scores, 2023.

[142] Jiashuo Liu, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui. Towards out-of-distribution generalization: A survey, 2023.

[143] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[144] Mingsheng Long, Yue Cao, Jianmin Wang, and Michael I. Jordan. Learning transferable features with deep adaptation networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 97–105. JMLR.org, 2015.

[145] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[146] Mario Lučić, Michael Tschannen, Marvin Ritter, Xiaohua Zhai, Olivier Bachem, and Sylvain Gelly. High-fidelity image generation with fewer labels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4183–4192. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/lucic19a.html.

[147] Spandan Madan, Tomotake Sasaki, Tzu-Mao Li, Xavier Boix, and Hanspeter Pfister. Small in-distribution changes in 3d perspective and lighting fool both CNNs and transformers. URL http://arxiv.org/abs/2106.16198.

[148] Spandan Madan, Timothy Henry, Jamell Dozier, Helen Ho, Nishchal Bhandari, Tomotake Sasaki, Frédo Durand, Hanspeter Pfister, and Xavier Boix. When and how convolutional neural networks generalize to out-of-distribution category–viewpoint combinations. *Nature Machine Intelligence*, 4(2):146–153, Feb 2022. ISSN 2522-5839. doi: 10.1038/s42256-021-00437-5. URL https://doi.org/10.1038/s42256-021-00437-5.

[149] Hartmut Maennel, Ibrahim Alabdulmohsin, Ilya Tolstikhin, Robert J. N. Baldock, Olivier Bousquet, Sylvain Gelly, and Daniel Keysers. What do neural networks learn when trained with random labels? In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

[150] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, pages 5188–5196. IEEE Computer Society, 2015. ISBN 978-1-4673-6964-0. URL http://dblp.uni-trier.de/db/conf/cvpr/cvpr2015.

html#MahendranV15.

[151] Shagufta Mehnaz, Sayanton V Dibbo, Ehsanul Kabir, Ninghui Li, and Elisa Bertino. Are your sensitive attributes private? novel model inversion attribute inference attacks on classification models. *arXiv preprint arXiv:2201.09370*, 2022.

[152] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint*, (arXiv:1411.1784), 2014.

[153] Samarth Mishra, Rameswar Panda, Cheng Perng Phoo, Chun-Fu Richard Chen, Leonid Karlinsky, Kate Saenko, Venkatesh Saligrama, and Rogerio S Feris. Task2sim: Towards effective pre-training and transfer from synthetic data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9194–9204, 2022.

[154] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

[155] Charlie Nash, Nate Kushman, and Christopher KI Williams. Inverting supervised representations with autoregressive neural density models. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1620–1629. PMLR, 2019.

[156] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 3395–3403, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[157] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017.

[158] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016.

[159] Mehdi Noroozi, Ananth Vinjimoor, Paolo Favaro, and Hamed Pirsiavash. Boosting self-supervised learning via knowledge transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9359–9367, 2018.

[160] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

[161] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724, 2014.

[162] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby,

Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023.

[163] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[164] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[165] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arxiv:2103.00020[cs]*, February 2021. doi: 10.48550/arXiv.2103.00020. URL `http://arxiv.org/abs/2103.00020`.

[166] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[167] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents, 2022.

[168] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML 2014)*, 2014.

[169] Robin Rombach, Patrick Esser, and Björn Ommer. Making sense of cnns: Interpreting deep representations and their invariances with inns. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pages 647–664. Springer, 2020.

[170] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. volume 9351, pages 234–241, 10 2015. ISBN 978-3-319-24573-7. doi: 10.1007/978-3-319-24574-4_28.

[171] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[172] Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization, 2022. URL `https://arxiv.org/abs/2202.06856`.

[173] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[174] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[175] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, Yann Ollivier, and Hervé Jégou. White-box vs black-box: Bayes optimal strategies for membership inference. In *International Conference on Machine Learning*, pages 5558–5567. PMLR, 2019.

[176] Shiori Sagawa*, Pang Wei Koh*, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=ryxGuJrFvS`.

[177] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, page 791–798, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595937933. doi: 10.1145/1273496.1273596. URL `https://doi.org/10.1145/1273496.1273596`.

[178] Ahmed Salem, Yang Zhang, Mathias Humbert, Pascal Berrang, Mario Fritz, and Michael Backes. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *arXiv preprint arXiv:1806.01246*, 2018.

[179] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *NeurIPS*, 2016.

[180] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P. Kingma. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

[181] Mert Bulent Sariyildiz, Karteek Alahari, Diane Larlus, and Yannis Kalantidis. Fake it till you make it: Learning transferable representations from synthetic imagenet clones. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[182] Mert Bülent Sarıyıldız, Yannis Kalantidis, Karteek Alahari, and Diane Larlus. No reason for no supervision: Improved generalization in supervised models. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=3Y5Uhf5KgGK`.

[183] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that? *arXiv preprint arXiv:1611.07450*, 2016.

[184] Shashank Shekhar, Florian Bordes, Pascal Vincent, and Ari Morcos. Objectives matter: Understanding the impact of self-supervised objectives on vision transformer

representations, 2023.

[185] Assaf Shocher, Yossi Gandelsman, Inbar Mosseri, Michal Yarom, Michal Irani, William T Freeman, and Tali Dekel. Semantic pyramid for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7457–7466, 2020.

[186] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2017.

[187] Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget, 2023.

[188] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[189] Amanpreet Singh, Ronghang Hu, Vedanuj Goswami, Guillaume Couairon, Wojciech Galuba, Marcus Rohrbach, and Douwe Kiela. FLAVA: A foundational language and vision alignment model. In *CVPR*, 2022.

[190] Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, and Martin Wattenberg. Smoothgrad: removing noise by adding noise. *arXiv preprint arXiv:1706.03825*, 2017.

[191] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 2256–2265. JMLR.org, 2015. URL `http://proceedings.mlr.press/v37/sohl-dickstein15.html`.

[192] Gowthami Somepalli, Vasu Singla, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Diffusion art or digital forgery? investigating data replication in diffusion models. *arXiv preprint arXiv:2212.03860*, 2022.

[193] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. 07 2019.

[194] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015.

[195] Oliver Struckmeier, Kshitij Tiwari, and Ville Kyrki. Autoencoding slow representations for semi-supervised data-efficient regression. *Machine Learning*, pages 1–19, 2023.

[196] Andrew Szot, Alex Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Chaplot, Oleksandr Maksymets, Aaron Gokaslan, Vladimir Vondrus, Sameer Dharur, Franziska Meier, Wojciech Galuba, Angel Chang, Zsolt Kira, Vladlen Koltun, Jitendra Malik, Manolis Savva, and Dhruv Batra.

Habitat 2.0: Training home assistants to rearrange their habitat, 2021.

[197] Kai Sheng Tai, Peter Bailis, and Gregory Valiant. Equivariant transformer networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6086–6095. PMLR, 2019. URL `http://proceedings.mlr.press/v97/tai19a.html`.

[198] Tristan Thrush, Ryan Jiang, Max Bartolo, Amanpreet Singh, Adina Williams, Douwe Kiela, and Candace Ross. Winoground: Probing vision and language models for visio-linguistic compositionality. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5238–5248, 2022.

[199] Nenad Tomasev, Ioana Bica, Brian McWilliams, Lars Buesing, Razvan Pascanu, Charles Blundell, and Jovana Mitrovic. Pushing the limits of self-supervised resnets: Can we outperform supervised learning without labels on imagenet? *arXiv preprint arXiv:2201.05119*, 2022.

[200] Brandon Trabucco, Kyle Doherty, Max Gurinas, and Ruslan Salakhutdinov. Effective data augmentation with diffusion models, 2023.

[201] Matthew Trager, Pramuditha Perera, Luca Zancato, Alessandro Achille, Parminder Bhatia, and Stefano Soatto. Linear spaces of meanings: Compositional structures in vision-language models, 2023.

[202] Trimble Inc. 3d warehouse. `https://3dwarehouse.sketchup.com/`. Accessed: 2022-03-07.

[203] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[204] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018.

[205] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *CVPR*, 2018.

[206] Asahi Ushio, Luis Espinosa Anke, Steven Schockaert, and Jose Camacho-Collados. BERT is to NLP what AlexNet is to CV: Can pre-trained language models identify analogies? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, Online, August 2021. Association for Computational Linguistics.

[207] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett,

editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/b1301141feffabac455e1f90a7de2054-Paper.pdf.

[208] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1747–1756. JMLR.org, 2016.

[209] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[210] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[211] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[212] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[213] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, page 1096–1103, New York, NY, USA, 2008. Association for Computing Machinery. ISBN 9781605582054. doi: 10.1145/1390156.1390294. URL https://doi.org/10.1145/1390156.1390294.

[214] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[215] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11 (110):3371–3408, 2010. URL http://jmlr.org/papers/v11/vincent10a.html.

[216] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European conference on computer vision (ECCV)*, pages 391–408, 2018.

[217] Tan Wang, Kevin Lin, Linjie Li, Chung-Ching Lin, Zhengyuan Yang, Hanwang Zhang, Zicheng Liu, and Lijuan Wang. Equivariant similarity for vision-language foundation models, 2023.

[218] Weiran Wang, Raman Arora, Karen Livescu, and Jeff Bilmes. On deep multi-view representation learning. In *International conference on machine learning*, pages 1083–1092. PMLR, 2015.

[219] Yizhou Wang, Shixiang Tang, Feng Zhu, Lei Bai, Rui Zhao, Donglian Qi, and Wanli Ouyang. Revisiting the transferability of supervised pretraining: an MLP perspective. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 9173–9183. IEEE, 2022. doi: 10.1109/CVPR52688.2022.00897. URL https://doi.org/10.1109/CVPR52688.2022.00897.

[220] Lauren Watson, Chuan Guo, Graham Cormode, and Alex Sablayrolles. On the importance of difficulty calibration in membership inference attacks. *arXiv preprint arXiv:2111.08440*, 2021.

[221] Lilian Weng. From autoencoder to beta-vae. *lilianweng.github.io*, 2018. URL https://lilianweng.github.io/posts/2018-08-12-vae/.

[222] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm, 2021.

[223] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[224] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[225] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *arXiv preprint arXiv:2006.09994*, 2020.

[226] Sirui Xie, Ari S Morcos, Song-Chun Zhu, and Ramakrishna Vedantam. COAT: Measuring object compositionality in emergent representations. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 24388–24413. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/xie22b.html.

[227] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, and Han Hu. Simmim: A simple framework for masked image modeling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9653–9663, 2022.

[228] Renzhe Xu Han Yu Zheyan Shen Peng Cui Xingxuan Zhang, Yue He. Nico++: Towards better benchmarking for domain generalization, 2022.

[229] Kaiyu Yang, Jacqueline Yau, Li Fei-Fei, Jia Deng, and Olga Russakovsky. A study of face obfuscation in imagenet. In *International Conference on Machine Learning (ICML)*, 2022.

[230] Jiayuan Ye, Aadyaa Maddi, Sasi Kumar Murakonda, Vincent Bindschaedler, and Reza Shokri. Enhanced membership inference attacks against machine learning models. *arXiv preprint arXiv:2111.09679*, 2021.

[231] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. Privacy risk in machine learning: Analyzing the connection to overfitting. In *2018 IEEE 31st computer security foundations symposium (CSF)*, pages 268–282. IEEE, 2018.

[232] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL `https://proceedings.neurips.cc/paper/2014/file/375c71349b295fbe2dcdca9206f20a06-Paper.pdf`.

[233] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[234] Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S Huang. Generative image inpainting with contextual attention. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5505–5514, 2018.

[235] Mert Yuksekgonul, Federico Bianchi, Pratyusha Kalluri, Dan Jurafsky, and James Zou. When and why vision-language models behave like bags-of-words, and what to do about it? In *International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=KRLUvxh8uaX`.

[236] Jure Zbontar, Li Jing, Ishan Misra, Yann Lecun, and Stephane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12310–12320. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/zbontar21a.html`.

[237] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[238] Hui Zeng and Xiaohui Cui. Simclrt: A simple framework for contrastive learning of rumor tracking. *Eng. Appl. Artif. Intell.*, 110:104757, 2022. doi: 10.1016/j.engappai.

2022.104757. URL https://doi.org/10.1016/j.engappai.2022.104757.

[239] Yan Zeng, Xinsong Zhang, and Hang Li. Multi-grained vision language pre-training: Aligning texts with visual concepts. *arXiv preprint arXiv:2111.08276*, 2021.

[240] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *International conference on machine learning*, pages 7354–7363. PMLR, 2019.

[241] Liheng Zhang, Guo-Jun Qi, Liqiang Wang, and Jiebo Luo. Aet vs. aed: Unsupervised representation learning by auto-encoding transformations rather than data. *arXiv preprint*, (arXiv:1901.04596), 2019.

[242] Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. In *ECCV*, 2016.

[243] Richard Zhang, Phillip Isola, and Alexei A. Efros. Split-brain autoencoders: Unsupervised learning by crosschannel prediction. In *CVPR*, 2017.

[244] Yuxuan Zhang, Huan Ling, Jun Gao, Kangxue Yin, Jean-Francois Lafleche, Adela Barriuso, Antonio Torralba, and Sanja Fidler. Datasetgan: Efficient labeled data factory with minimal human effort. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10145–10155, 2021.

[245] Nanxuan Zhao, Zhirong Wu, Rynson W. H. Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=tC6iW2UUbJf.

[246] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014. URL https://proceedings.neurips.cc/paper/2014/file/3fe94a002317b5f9259f82690aeea4cd-Paper.pdf.

[247] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. In *ICLR*, 2022.

[248] Pan Zhou, Yichen Zhou, Chenyang Si, Weihao Yu, Teck Khim Ng, and Shuicheng Yan. Mugs: A multi-granular self-supervised learning framework. 2022.

Appendix A

High Fidelity Visualization of What Your Self-Supervised Representation Knows About

A.1. Conditional and super-resolution sampling with RCDM

As presented in the main text, we introduce RCDM to generate samples that preserved well the semantics of the images used for the conditioning. The training of the model is simple and presented in Figure 1b. We show in Figure 3 additional samples of RCDM when conditioning on the SSL representation of ImageNet validation set images (which were never used for training). We observe that the information hidden in the SSL representation is so rich that RCDM is almost able to reconstruct entirely the image used for conditioning. To further evaluate the abilities of this model, we present in Figure 4 a similar experiment except that we use out of distribution images as conditioning. We used cell images from microscope and a photo of a status (Both from Wikimedia Commons), sketch and cartoons from PACS [137], image segmentation from Cityscape [54] and an image of the Earth by NASA. Even in the OOD scenario, RCDM is able to generate images that share common features to the one used as conditioning because of the richness of ssl representations. However, if the images used as OOD are too far from the training distribution, which is the case when using image segmentation mask from Cityscapes, the model will have more difficulty to reconstruct the images used as conditioning. To investigate if this failure is due to the SSL network used to produced the conditioning, we run the experiment in Figure 6 in which we kept the same SSL model with an RCDM trained on ImageNet and another one on Cityscape. We observe that when using Cityscapes segmentation mask as conditioning with an RCDM trained on Cityscapes segmentation mask, despite using a SSL model trained only on ImageNet, RCDM is able to reconstruct the conditioning very faithfully. This mean that the failure mode

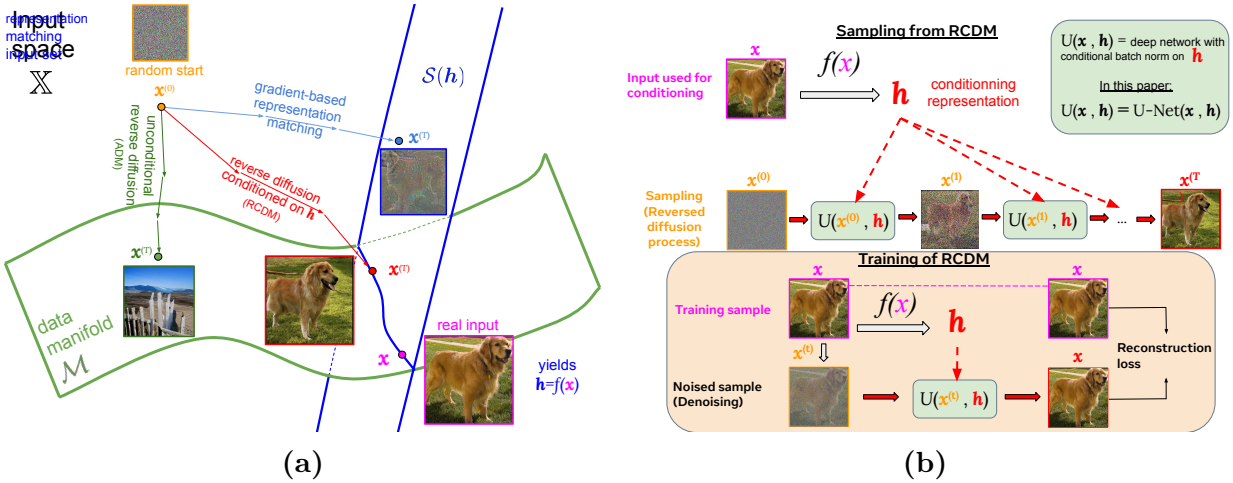


Fig. 1. (a) Illustration of considered image generation methods. A real input $\boldsymbol{x}$ yields representation $\boldsymbol{h}$. All methods start from a random noise image $\boldsymbol{x}^{(0)}$. Gradient-based representation matching (light blue arrows) will move it towards $\mathcal{S}(\boldsymbol{h})$ i.e. until its representation matches $\boldsymbol{h}$, but won't land on the data-manifold $\mathcal{M}$. Unconditional reverse diffusion (ADM model, green arrows) will move it towards the data manifold. Our representation-conditioned diffusion model (RCDM, red arrows) will move it towards $\mathcal{M} \cap \mathcal{S}(\boldsymbol{h})$, yielding a different natural-looking image with the same given representation. (b) Representation-Conditionned Diffusion Model (RCDM). From a diffusion process that progressively corrupts an image, the model learns the reverse process by predicting the noise that it should remove at each step. We also add as conditioning a vector $\boldsymbol{h}$, which is the representation given by a SSL or supervised model for a given image $\boldsymbol{x}$. Thus, the network is trained explicitly to denoise towards a specific example given the corresponding conditioning. The diffusion model used is the same as the one presented by Dhariwal and Nichol [60] with the exception of the conditioning on the representations.

observed in OOD are mostly due to the visualization model (RCDM) and not due to the representation used for conditioning.

In those examples we used conditional batch-normalization (which is the same technique as used by Casanova et al. [46]). However one can also use the sampling technique built-in in the ADM model of Dhariwal and Nichol [60]. Instead of using an embedding layer that take discrete representation, we can use a linear layer to map a representation to the dimension of the time steps embedding and add it along the time step conditioning. A comparison with these two conditioning methods is shown in Figure 5.

We also use the super-resolution model presented by Dhariwal and Nichol [60] to generate images of higher resolutions. In Figure 8, we use the small images on the top of the bigger images as conditioning for a RCDM trained on images of size 128x128. Then, we feed the 128x128 samples into the super-resolution model of Dhariwal and Nichol [60] to get images of size 512x512. Since the model of Dhariwal and Nichol [60] is conditional and need labels, we used a random label when upsampling from RCDM. Despite using the "wrong" label, the high resolution samples are still very close to the conditioning. This show that RCDM can

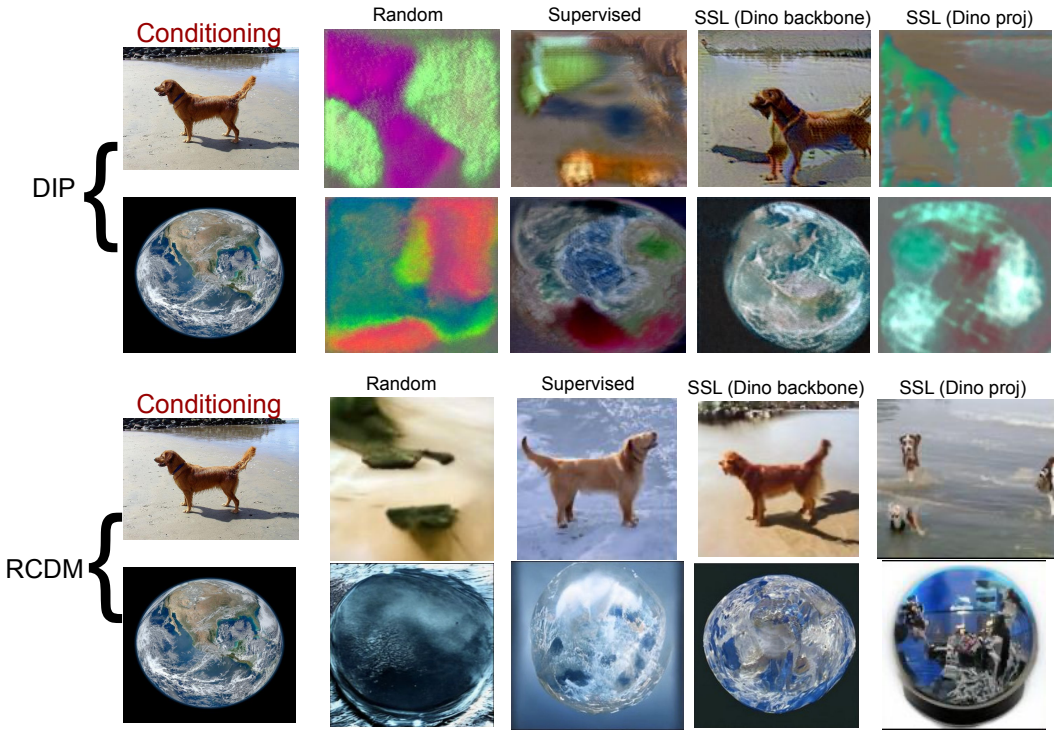


Fig. 2. We compare RCDM with the approach of Zhao et al. [245] that use Deep Image Prior (DIP [203]) to visualize the features learn by Self-supervised models. We run this experiment by using as conditioning an In-Distribution image (with the dog from ImageNet validation set) and an Out-Of-Distribution image with an image from the Earth (Source: NASA). For both methods, we compare with representations extracted at the backbone level of a Resnet (after average pooling) from a random initialized network, a supervised network and a network trained with SSL (Dino). We also use the representation extracted at the projector level for Dino. We observe that the samples we obtained with RCDM have a better quality than the ones generated with DIP and are also more insightful about the properties of the representations. In this instance, we clearly see with RCDM that the supervised representation is invariant to background which is something more difficult to assess with DIP.

be used jointly with a super-resolution model to sample high fidelity images in the close neighborhood of the conditioning.

To verify how well our model can produce realistic samples from different combinations of representations, we take two images from which we compute their representations and perform a linear interpolation between those. This give us new vectors of representation that can be used as conditioning for RCDM. We can see on Figure 9 and Figure 10 that RCDM is able to generate samples that contains the semantic characteristics of both images.

Finally, in Figure 11, we search the nearest neighbors of a series of samples in the ImageNet training set. As demonstrated by Figure 11, RCDM samples images that are new and far enough from images belonging to the training set of ImageNet.

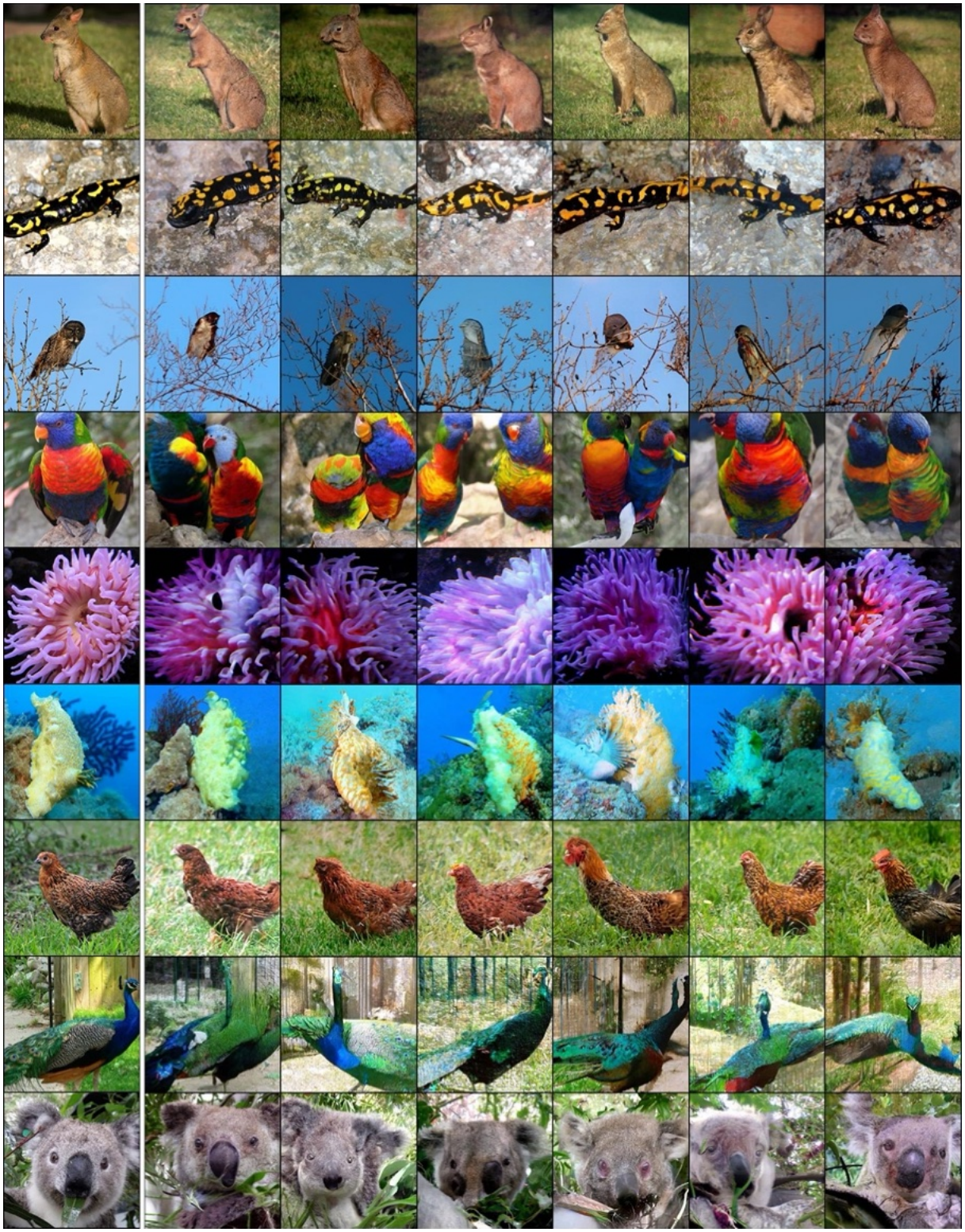


Fig. 3. Generated samples from RCDM on 256x256 images trained with representations produced by Dino. We put on the first column the images that are used to compute the representation conditioning. On the following column, we can see the samples generated by RCDM. It is worth to denote our generated samples are qualitatively close to the original image.

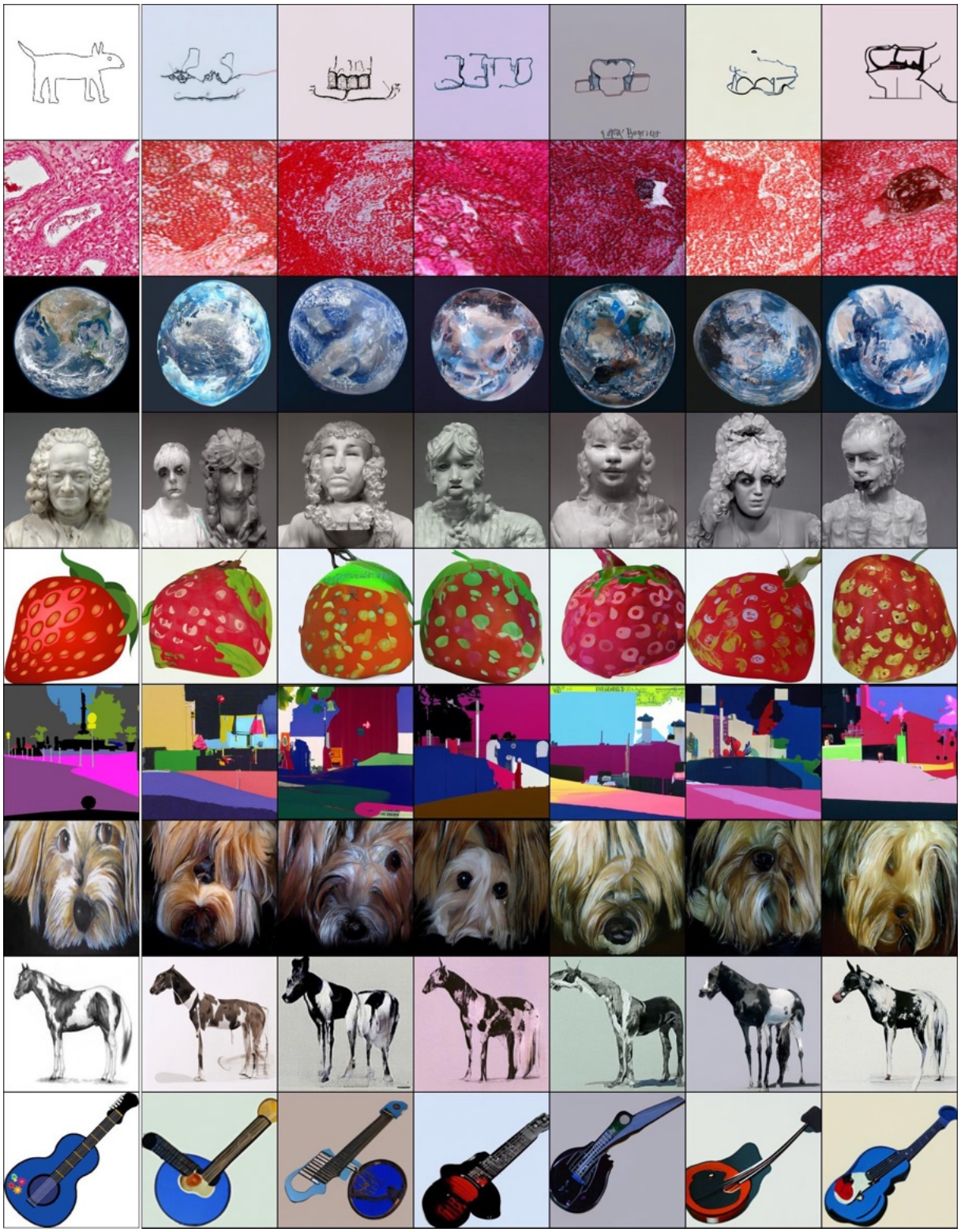


Fig. 4. Generated samples from RCDM model on 256x256 images trained with representations produced by Dino on Out of Distribution data. We put on the first column the images that are used to compute the representation. On the following column, we can see the samples generated by RCDM. It is worth to denote our generated sample are close to the original image. The images used for the conditioning are from Wikimedia Commons, Cityscapes [54], PACS [137] and the image of earth from NASA.

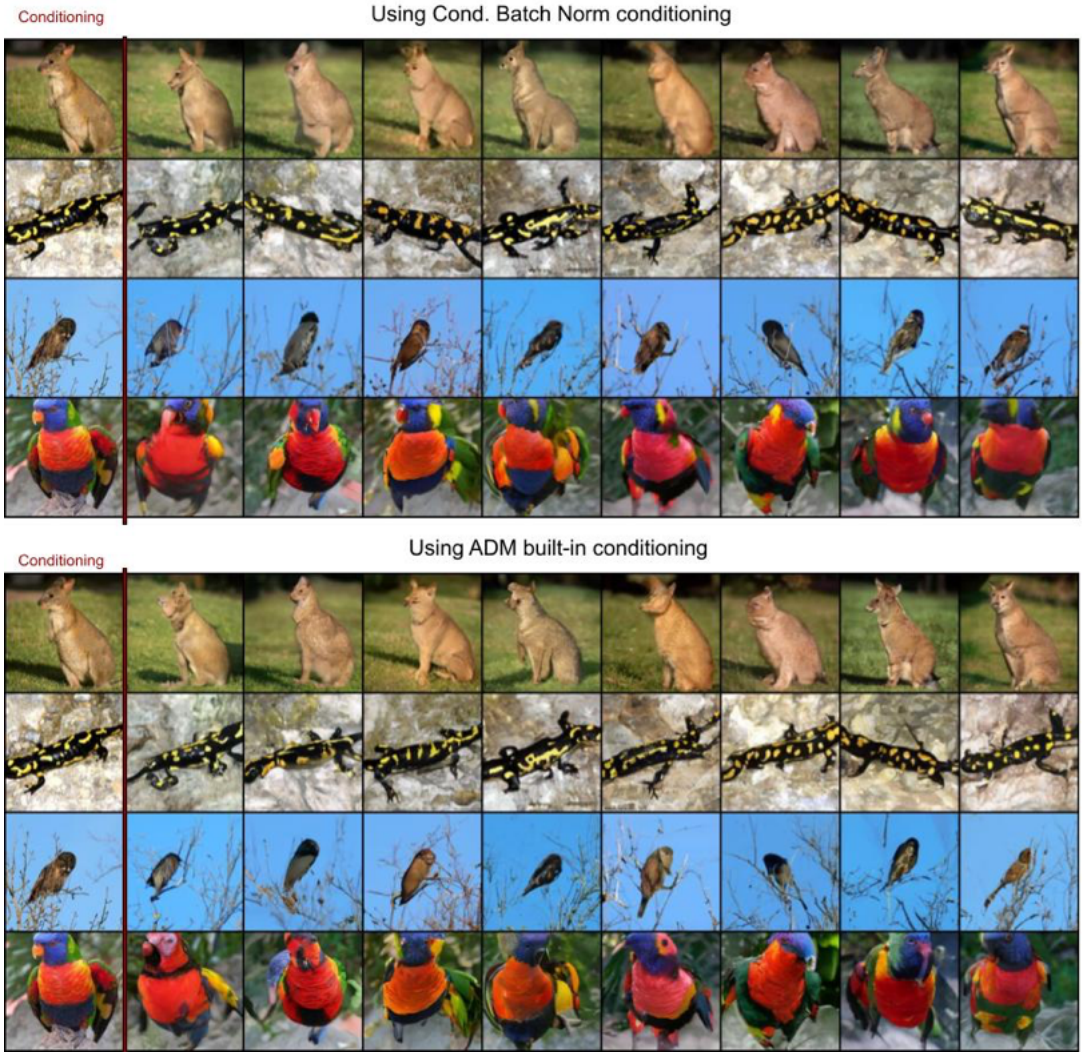


Fig. 5. Comparison between conditioning RCDM with batch normalization and the built-in conditioning mechanism offered by ADM. For this example, we took the representation backbone of dino trained on ImageNet with resolution 128x128. There doesn't seem to be any significant differences between both methods.

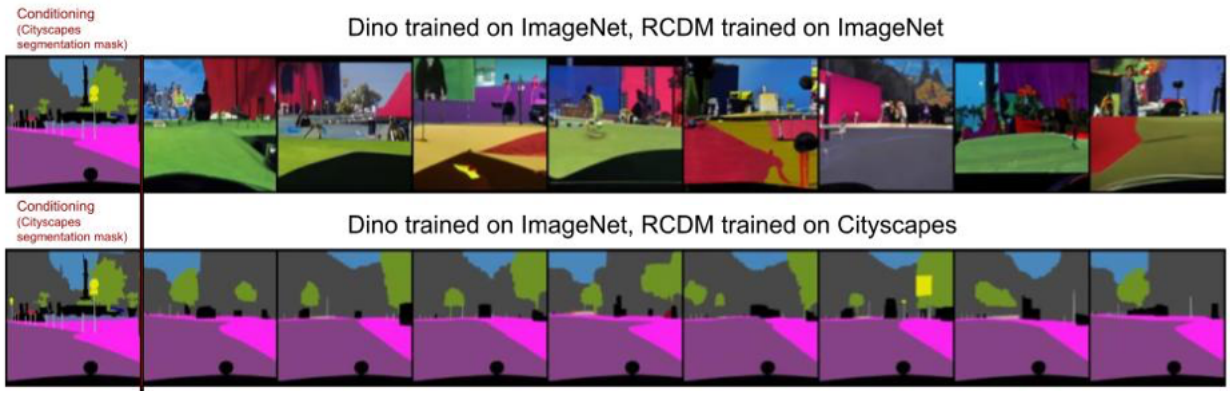
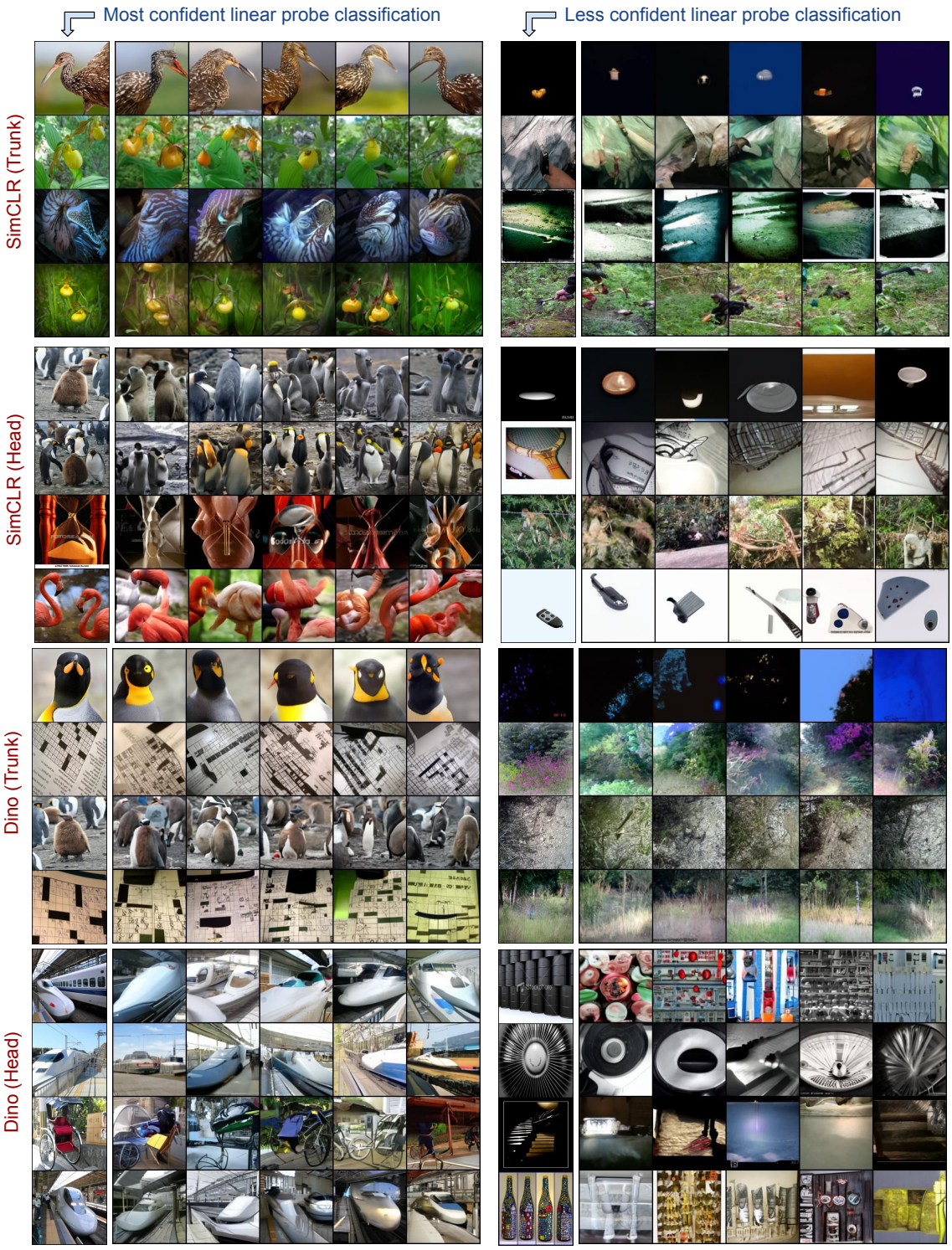
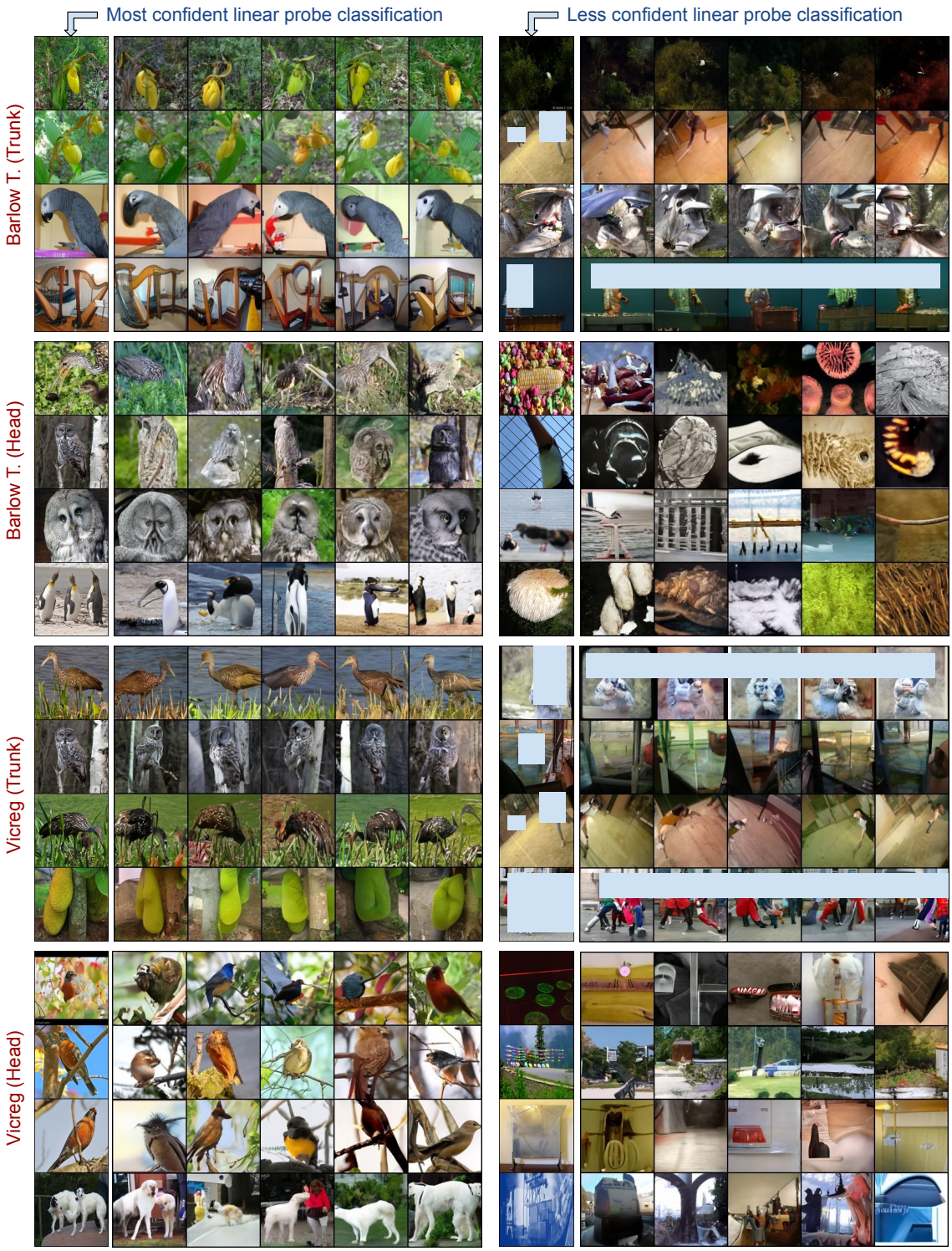


Fig. 6. We perform this experiment to see if the failures mode on OOD, especially when conditioning on segmentation mask of Cityscapes, are due to the self-supervised representations not containing enough information to reconstruct the image, or are due to RCDM not being able to reconstruct OOD images. On the first line, we show the samples generated by an RCDM trained on ImageNet with the self-supervised representation of Dino that was also trained on ImageNet. On the second line, we show the samples generated by an RCDM trained on the segmentation masks of cityscapes that use the same self-supervised model of Dino that was trained on ImageNet. We can clearly see that despite using a SSL model trained on ImageNet, when RCDM is trained on CityScapes, the reconstruction almost match the original conditioning. Hence, one should train or fine-tune RCDM on any target dataset to then use it to sample representation conditioned images from a (frozen) pre-trained model.

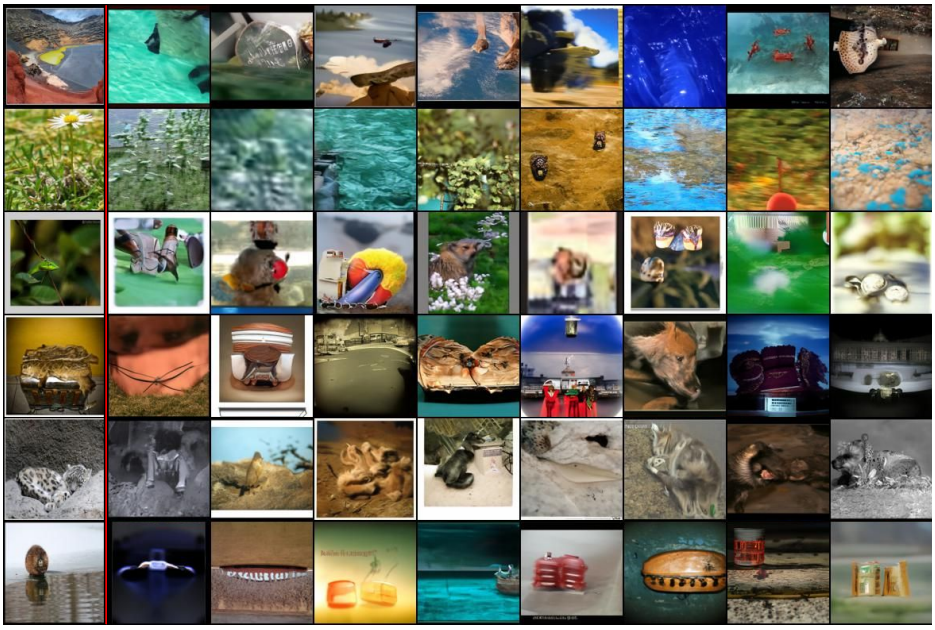


a) RCDM samples using a conditioned-representation having the highest probabilities under a linear classifier.



b) RCDM samples on conditioning that have the lowest probabilities under a linear classifier.

c) More RCDM samples on conditioning that have the highest probabilities under a linear classifier.



d) More RCDM samples on conditioning that have the lowest probabilities under a linear classifier.

Fig. 7. We trained a linear probe for classification by using the representation (at the backbone and projector level) given by various SSL models. Then we find, among the ImageNet validation set, the images that yield the highest softmax probability under this linear probe and use RCDM to generate samples with respect to their representations. We also find the images that yield the lowest softmax probabilities. On the first column and seventh column in a) and b), we present the images used as conditioning (thus, the ones with highest and lowest softmax probabilities). In the following columns we show the corresponding RCDM samples. We observe that all generated images belong to the same class as the one that was used as conditioning when looking at the most confident representation . However when looking at the least confident representation, the generated samples does not seem to belong to a unique class. This experiment shows that the uncertainty in the predictions of a downstream classification task can also be predicted by simple visual inspection of samples produces by RCDM.

A.2. A hierarchical diffusion model for unconditional generation

We provided a novel and conditional generative model based on a given latent representation e.g. from a SSL embedding, and a diffusion model. This allows visualizing and thus provides insight regarding what is or isn't encoded in a particular representations. We can go one step further and augment this conditional model with an unconditional one that can generate those representations. This will provide us with the ability to generate new samples without the need to condition on a given input. As a by-product, it will allow us to quantify the quality of our generative process in an unconditional manner to fairly compare against state-of-the-art generative models.

We shall recall that our goal is to employ the conditional generative model to provide understanding into learned (SSL) representations. The unconditional model is only developed to compare our generative model and ensure that its quality is reliable for any further down analysis. As such, we propose to learn the representation distribution in a very simple manner via the usual Kernel Density Estimation (KDE). That is, the distribution is modeled as

$$p(\boldsymbol{h}) = \frac{1}{N} \sum_{n=1}^{N} \mathcal{N}(\boldsymbol{h}; f(\boldsymbol{x}_n); I\sigma)$$

with σ set to 0.01. By using the above distribution, we are able to sample representations $\boldsymbol{h}$ to then sample images $\boldsymbol{x}$ conditionally to that $\boldsymbol{h}$ using our diffusion model. We provide some samples in Figure 15 to show that even with our very simple conditioning, our method is still able to generate realistic images.

A.3. On the closeness of the samples in the representation space

Even if we show that RCDM is able to generate images that seems visually close to the image used for the conditioning, it's still unclear how close those images are in the representation space. We can compute euclidean distances but to know how close the generated samples are to the conditioning, we need to have references that can be used to compare this distance with. As references, we compute the euclidean distance between a conditioning image and random images in the validation set of ImageNet, random images belonging to the same class as the conditioning, the closest images in the training set, the conditioning image on which we applied single data augmentations and the conditoning image on which we applied the data augmentation performed by Swav and Dino [40, 44]. The results can be seen in Figure 16 for a RCDM trained with Dino representations and in Figure 17 for a RCDM trained with SimCLR representations. On both Figure, we observe that the

generated images with RCDM are closer to the conditioning than the closest neighbors in the entire training set of ImageNet. We also computed the mean and reciprocal mean rank in the main paper (Table 1b) which show that for most SSL models the closest examples in the representation space of the generated images is the image used as conditioning. We also added Figure 12 to show which rank is associated to samples generated by RCDM. For SimCLR, the rank is mostly always 1 whereas we got more diversity for the supervised case. This difficulty of RCDM to generated samples which have their representation that map back to the one used for the conditioning can be explain by the nature of a supervised training. In such scenario, the encoder is trained to map a big set of images (often a specific class) to a specific type of representation whereas SSL models are explicitly train to push each examples farther away from each others. Thus, it seems more likely that a little perturbation on the supervised representation induces a change of nearest neighbor. This hypothesis is supported by Figure 24 which show that small adversarial attack are enough to induces a change of class in the representation which is not the case for SSL encoders.

A.4. Analysis of representations learned with Self-Supervised model

Having generated samples that are close in the representation space to a conditioning image can give us an insight on what's hidden in the representations learned with self-supervised models. As demonstrated in the previous section, the samples that are generated with RCDM are really close visually to the image used as conditioning. This give an important proof of how much is kept inside a SSL representation. However, it's also important to consider how much this amount of "hidden" information varied depending on the SSL representation that is used. Therefore, we train several RCDM on SSL representations given by VicReg [21], Dino [44], Barlow Twins [236] and SimCLR [49]. In many applications that used self-supervised models, the representation that is used is the one corresponding to the backbone of the ResNet50. Usually, the representation given by the projector of the SSL-model (on which the SSL criterion is applied) is discarded because the results on many downstream tasks like classification is not as good as the backbone. However, since our work is to visualize and better understand the differences between SSL representations, we also trained RCDM on the representation given by the projector of Dino, Barlow Twins and SimCLR. In Figure 19 we condition all the RCDM with the image labelled as conditioning and sample 9 images for each model. If we look at the projector of the SSL models, the generated samples have a higher variance.

To further compare and analyse the different SSL models, we visualize how much SSL representations can be invariant with respect to a transformation that is applied on the conditioning image. In Figure 20, we apply several Data Augmentation: Vertical shift, Zoom

out, Zoom In, Grayscale and a Collor Jitter on a given conditioning image. Then we compute the SSL representations of the transformed image with different SSL models and use our corresponding RCDM to see how much the samples have changed with respect to the samples generated on the vanilla conditioning image. We observe that the representation (the 2048 backbone one) of all SSL methods are not invariant to scale and change of colors. Whereas the representation of the projector doesn't seem to take into account any small transformation in the original conditioning outside the scale for Dino. For SimCLR, there is still some information about the background that is kept in the representation however the samples are not as close visually with respect to the 2048 representation. Barlow Twins is interesting because there isn't much differences between the backbone representation (2048) one and the representation of the projector (Size 8192). With the exception that this last representation seems to be more invariant to color shift than the backbone one.

We also perform an experiment in Figure 23 using OOD images to ensure that the conclusions drawn with our methods about SSL representation are not specific to ImageNet.

A.4.1. Visualization of adversrial examples

We use RCDM to visualize adversarial examples for different models. For each model, we trained a linear classifier on top of their representations to predict class labels for the ImageNet dataset. Then, we use FGSM attacks over the trained model using a NLL loss to generate adversarial examples. In Figure 24 we show the adversarial examples that are created for each model, the samples generated by RCDM with respect to the representation of the adversarial perturbed example and the class label predicted by the linear classifier over the adversarial examples. The supervised model is very sensitive to the attack whereas SSL models seems more robust.

A.4.2. Manipulation of SSL representations

It is also possible to manipulate SSL representations to generate new images. We try to apply addition and subtractions over SSL representations (similarly to what has been done in NLP). From two different images, we compute the difference between the two corresponding representations and add the difference vector to a third image. Figure 27 shows that it is possible to apply such transformations meaningfully in the SSL space. We also used another setup where we choose specific dimensions in the representation based on how many times these dimensions are non zero in the representation space of a set of neighbors. Then we set this dimension to zero which surprisingly induces the removing of the background in the generated images. We also replace them by the same corresponding dimension of another images which induces a change of background toward the one of the new image. Results are shown in Figure 25.

A.4.3. Experiments with vision transformers

All the experiments in this paper were conducted with Resnet50 since most of the SSL baselines are available with this model. However, RCDM can work with any type of architecture, including vision transforms. In Figure 28, we show RCDM samples using representations of Dino trained with a VIT-B 16 [126].

A.4.4. Why is my model over-fitting on the training set ?

By enabling the visualization of what is learned in a representation, RCDM can help researchers to get a better understanding of the failures modes of their models. In one of our experiments, we trained a SSL models with VicReg by using only cropping as data augmentation (thus discarding the traditional colorjiterring/grayscaling and other transforms that change the colors). Training a linear probe on such network resulted in a training accuracy of 95% on the training set while the validation accuracy was only about 20%. To better understand how the model was able to overfit on the training set, we trained RCDM on the representations of this model. The samples obtained are shown in Figure 29. This experiment validate the hypothesis that removing color related augmentations during the training of SSL models leads to learn representations that are only colors and textures based.

A.4.5. Visualizing how representations are changing during training

Another way one can use RCDM, is to consider how representations are changing during training. In this experiment, we trained 3 RCDM models on the representation given by SSL models (VicReg) trained after 1 epoch, 5 epochs and 50 epochs. In this experiment, we want to visualize what is changing in the representation during training. The hypothesis was that at the beginning of the training, the network is learning some easy feature, like some color information, and later in the training more complex features, probably containing more shape based information. In Figure 30, we observe that after 1 epoch of SSL training, the information retain in the representation is mostly color/texture based while after only 5 epochs, we can see that the shape are better defined.

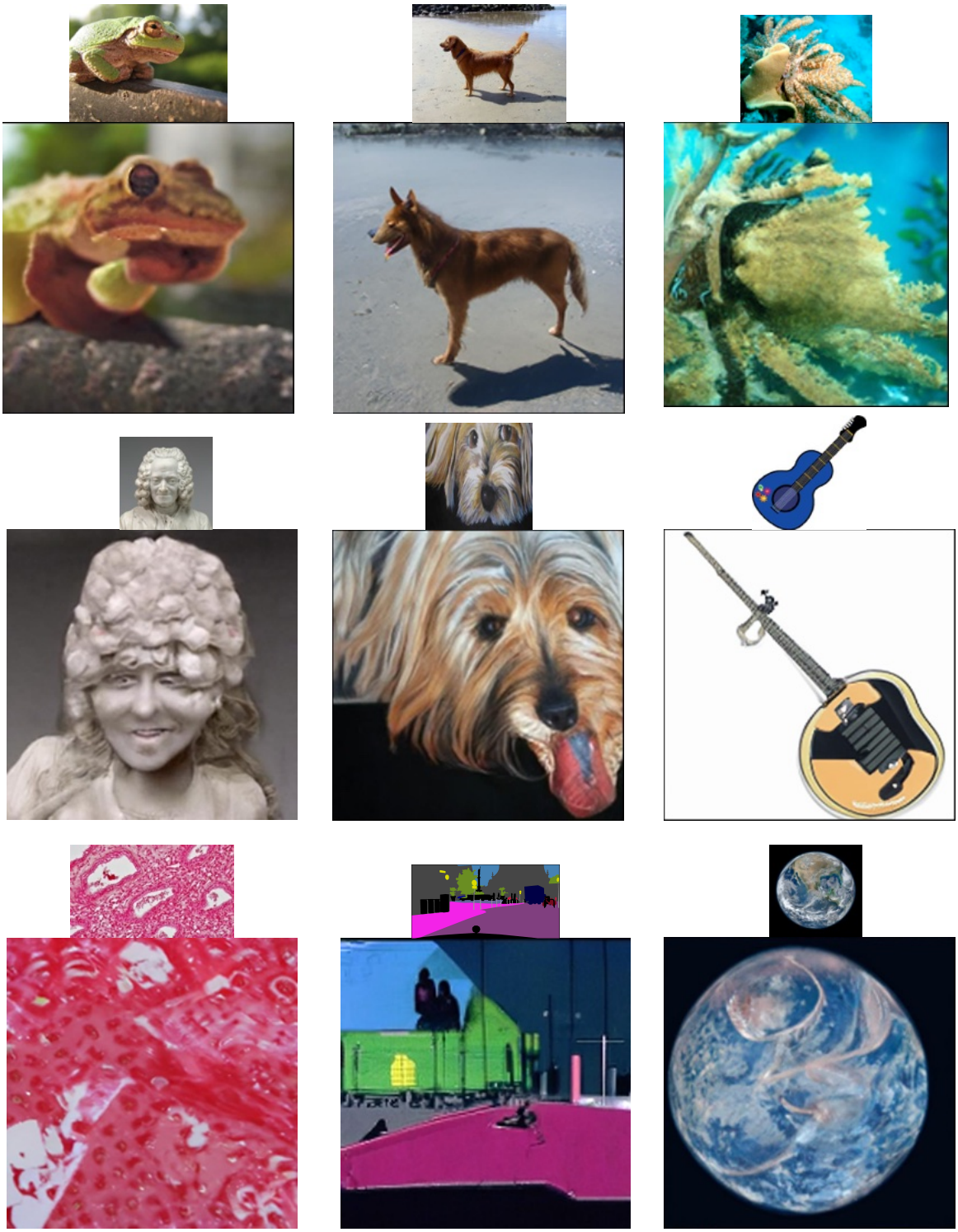
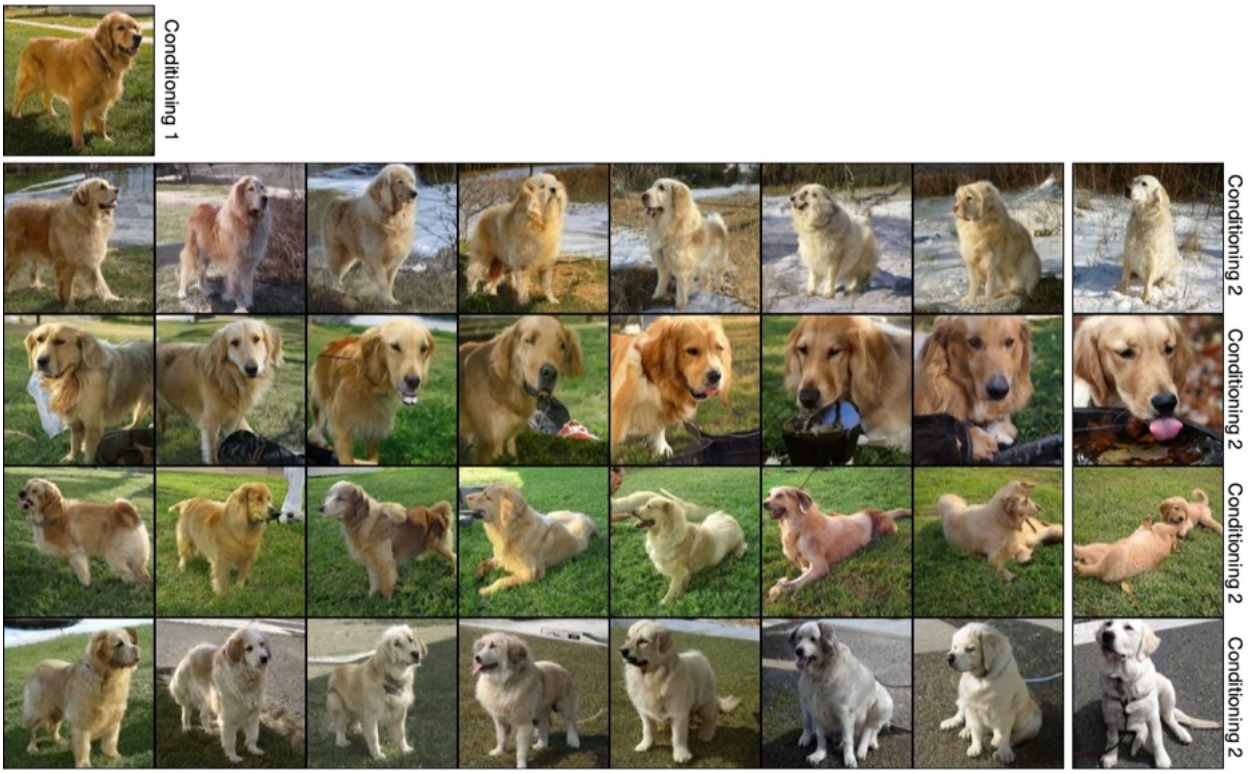
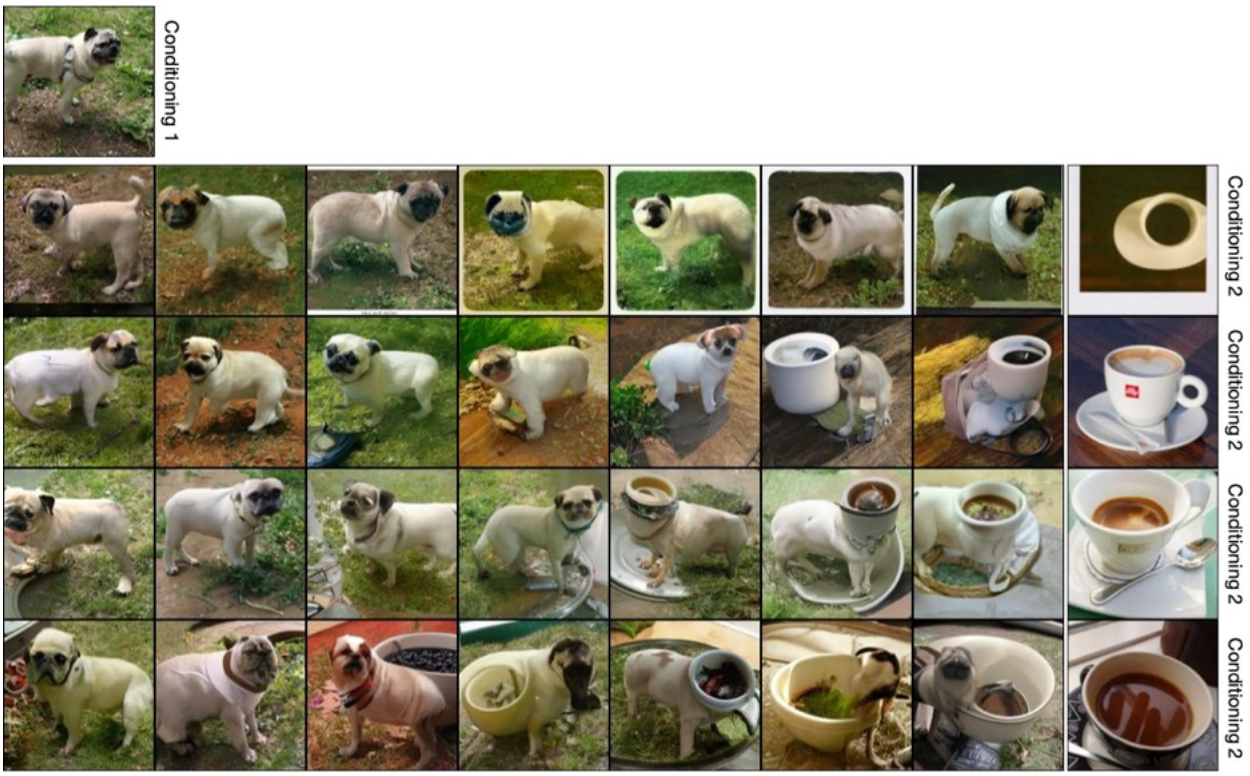


Fig. 8. High resolution samples from our conditional diffusion generative model using the super resolution model of Dhariwal and Nichol [60]. We use the small images on the top of each bigger image as conditioning (source NASA for the earth picture) for a diffusion model trained with Dino representation on 128x128 images. Then, we feed the samples generated to the super resolution model of Dhariwal and Nichol [60] which produces images of size 512x512. Since the super resolution model is conditional, we sample a random label. We note that the high resolution samples are still very close to the conditioning.



(a) Linear interpolation between the image of the golden retriever in conditioning 1 with various other images belonging to the same class as conditioning 2.



(b) Linear interpolation between the image of the pug in conditioning 1 with various other images belonging to the espresso class as conditioning 2.

Fig. 9. Each vectors that result from the linear interpolation is feed to a RCDM trained with Barlow Twins representation.

Fig. 10. Diversity of the samples generated by RCDM on interpolated representations. Each row corresponds to different random noise for the same conditioning. On the first and last column are the real images used for the interpolation. All of the images in-between those rows are samples from RCDM.

(a) Closest real images (ImageNet training set) from images sampled with RCDM trained on Dino (backbone) representation (2048).

(b) Closest real images (ImageNet training set) from images sampled with RCDM trained on Dino projector representation (256).

Fig. 11. We find the nearest neighbors in the representation space of samples generated by RCDM. The images in the red squared are the ones used for conditioning.

Fig. 12. After generating samples with respect to a specific conditioning, we compute back the representation of the generated samples and find which are the closest neighbors in the validation set. Then, we compute the rank of the original image that was used as conditioning within the set of neighbors. When the rank is one, it implies that the nearest neighbors of the generated samples is the conditioning itself, meaning that the generated samples have their representation that is very close in the representation space to the one used as conditioning. We can see that for SimCLR, the generated samples are much closer in the representation space to their conditioning than the supervised representation. This is easily explain by the fact that supervised model learn to map images from a same class toward a similar representation whereas SSL models try to push further away different examples.

Fig. 13. Visual analysis of the variance of the generated samples for a specific image when using the trunk/backbone of a Barlow Twins encoder. The first image (in red) in the one used as conditioning.

Fig. 14. Visual analysis of the variance of the generated samples for a specific image when using the projector/head of a Barlow Twins encoder. The first image (in red) in the one used as conditioning.

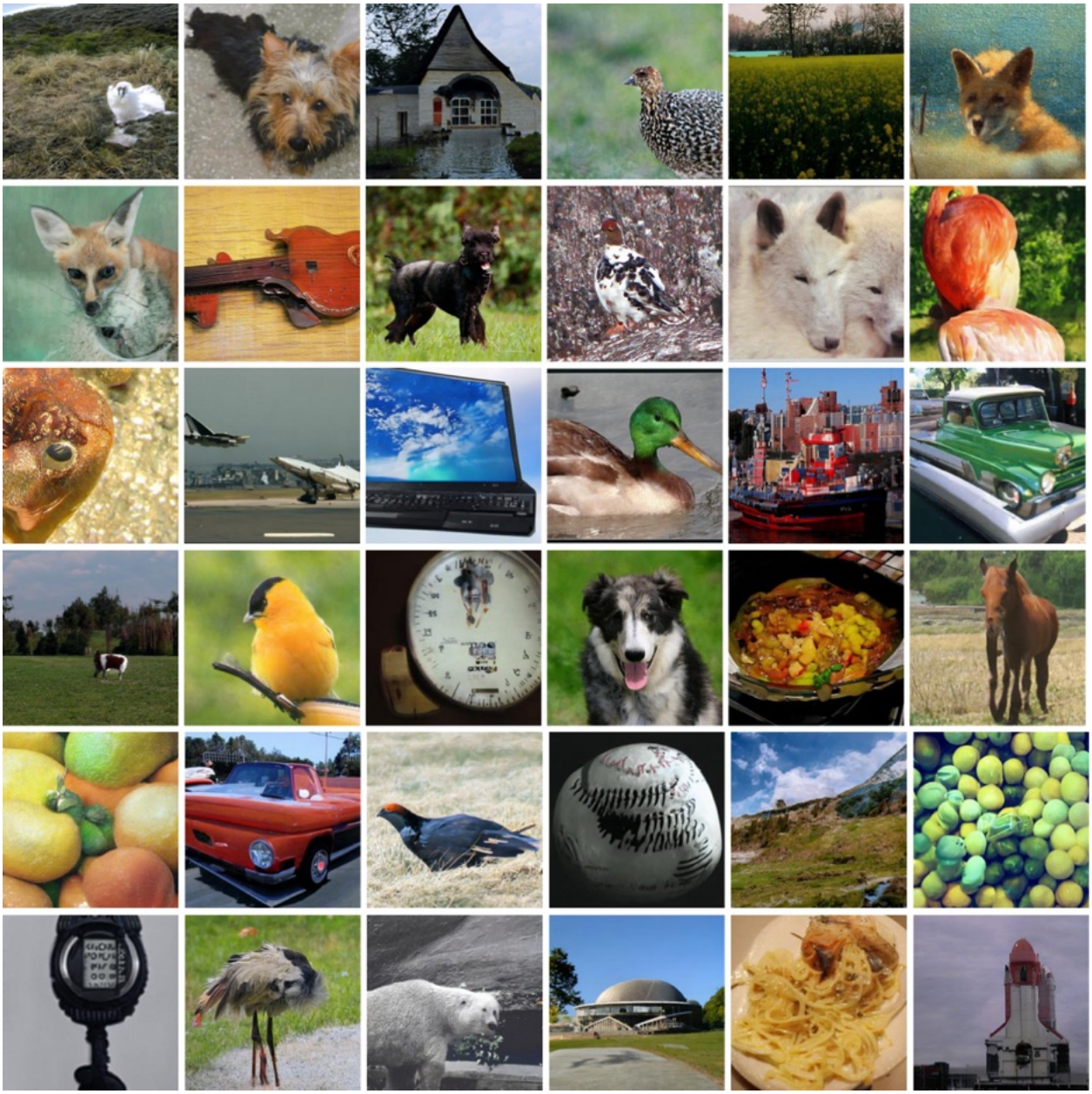


Fig. 15. Unconditional generation following the protocol of section A.2. Our simple generative model of representations consists in applying a small Gaussian noise over representation computed from random training images of ImageNet. We use these noisy vector as conditioning for our 256x256 RCDM trained with Dino representations. We note that the generated images looks realistic despite some generative artefact like a two-headed dog and an elephant-horse.

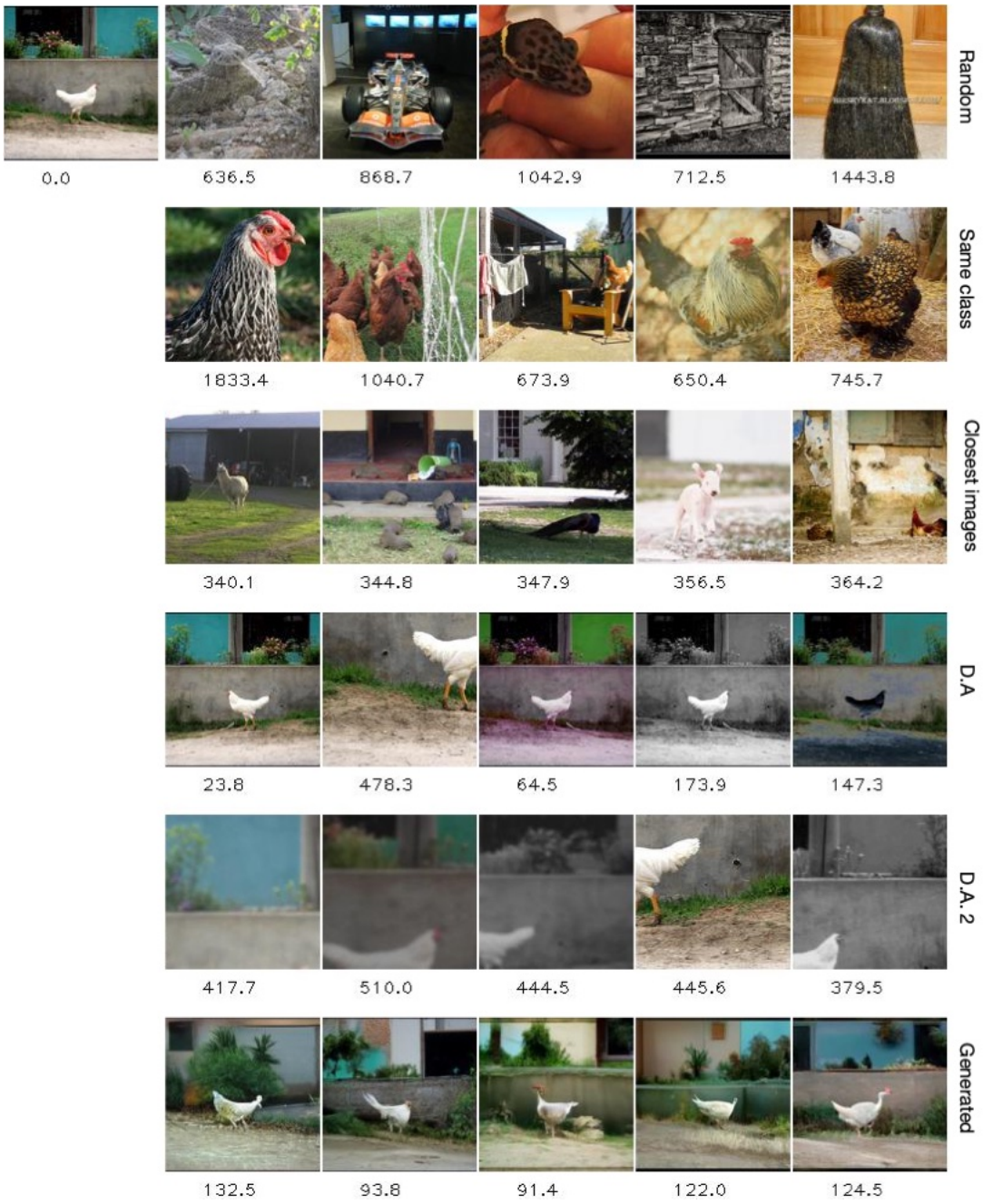


Fig. 16. Squared Euclidean distances in the Dino representation space. We show the squared euclidean distance between the conditioning image on the leftmost column on first row and different images to get an insight about how close the samples generated by the diffusion model stay close to the representation used as conditioning. The distances with the conditioning is printed below each images. On the first row, we show random images from the ImageNet validation data. On the second row, we take random validation examples belonging to the same class as the conditioning. On the third row, we find the closest training neighbors of the conditioning in the representation space. On forth row, D.A. means Data Augmentation which consist in horizontal flip, CenterCrop, ColorJitter, GrayScale and solarization. On fifth row (D.A. 2), we use the random data Augmentation used in the paper of [40, 44]. On the last row, we show the generated samples from our conditional diffusion model that use **Dino representation**. The samples produces by our model are much closer to the conditioning than other images.

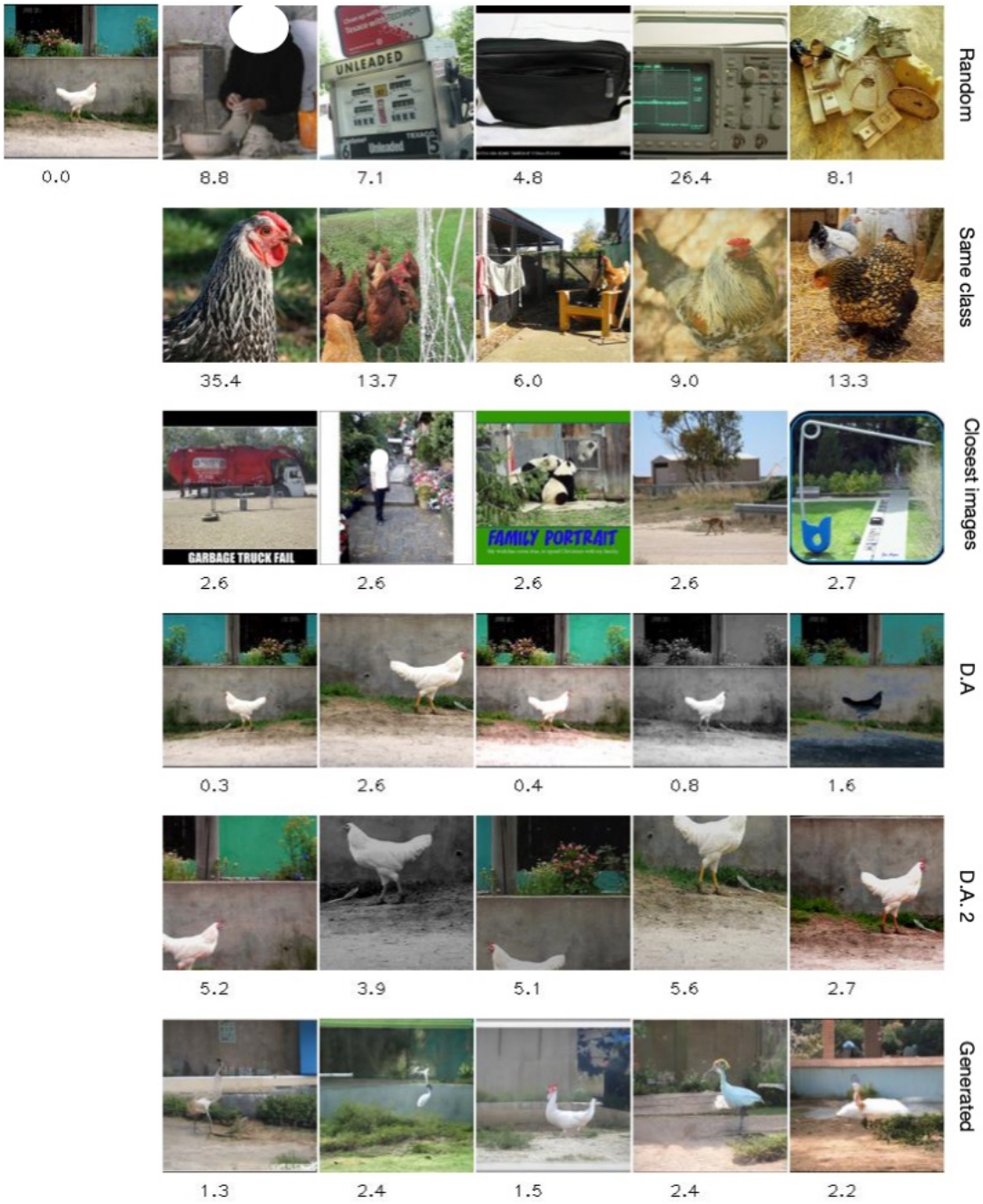


Fig. 17. Squared Euclidean distances in the SimCLR projector head representation space. We show the squared euclidean distance between the conditioning image on the leftmost column on first row and different images to get an insight about how close the samples generated by the diffusion model stay close to the representation used as conditioning. The distances with the conditioning is printed below each images. On the first row, we show random images from the ImageNet validation data. On the second row, we take random validation example belonging to the same class as the conditioning. On third row, we find the closest training neigbords of the conditioning in the representation space. On forth row, D.A. means Data Augmentation which consist in horizontal flip, CenterCrop, ColorJitter, GrayScale and solarization. On fifth row (D.A. 2), we use the random data Augmentation used in the paper of [40, 44]. On the last row, we show the generated samples from our conditional diffusion model that use **SimCLR projector head representation**. The samples produces by our model are much closer to the conditioning than other images.

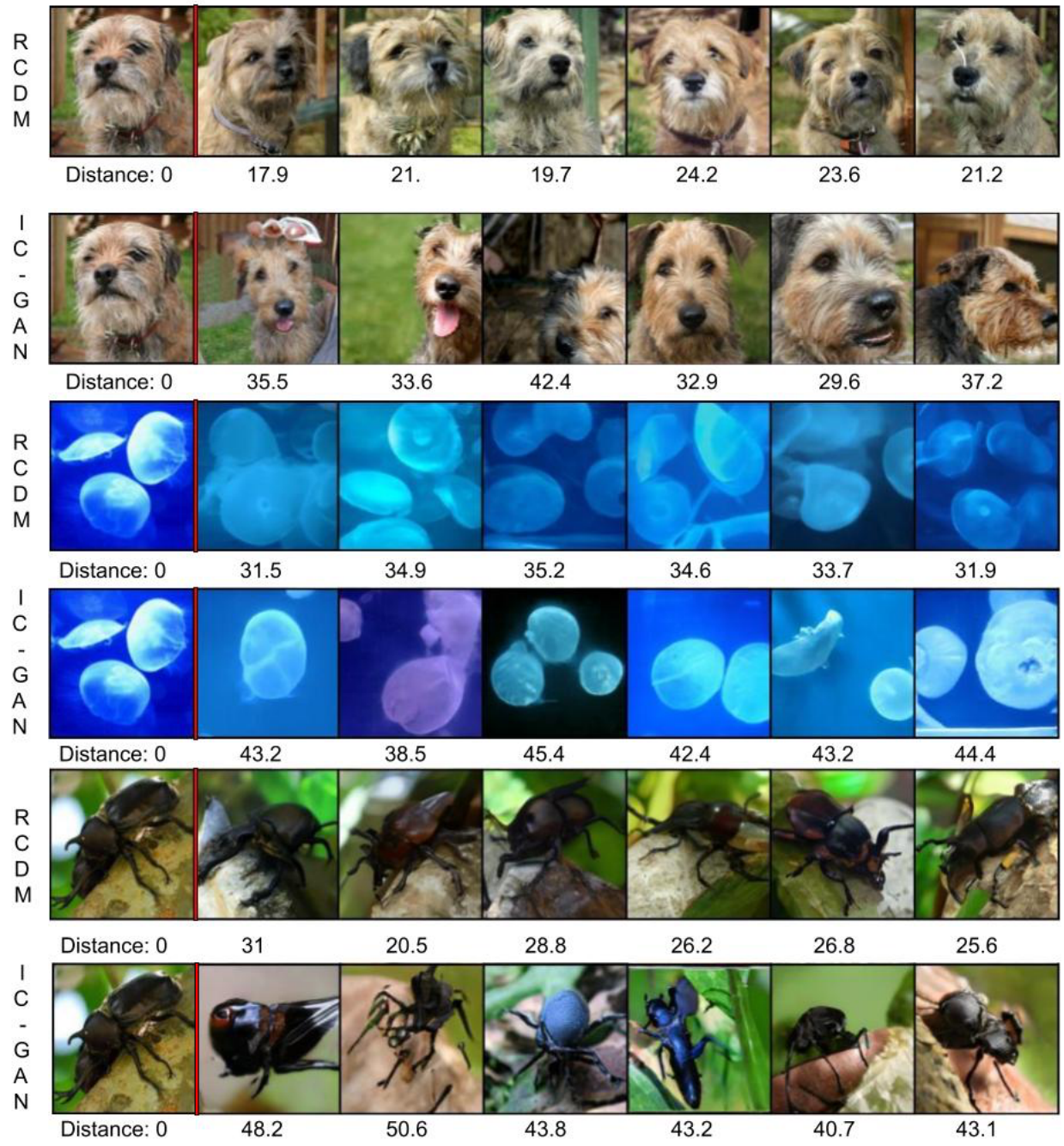


Fig. 18. Comparison of the euclidean distance between IC-GAN and RCDM. We use the same self-supervised representation as conditioning (Swav encoder) for RCDM and IC-GAN. We compute the euclidean distance between the representation of the generated images versus the representation used as conditioning. We observe that samples of RCDM are much closer in the representation space (and also visually) to the conditioning. Samples of IC-GAN show a higher variability, thus farther away in the representation space.

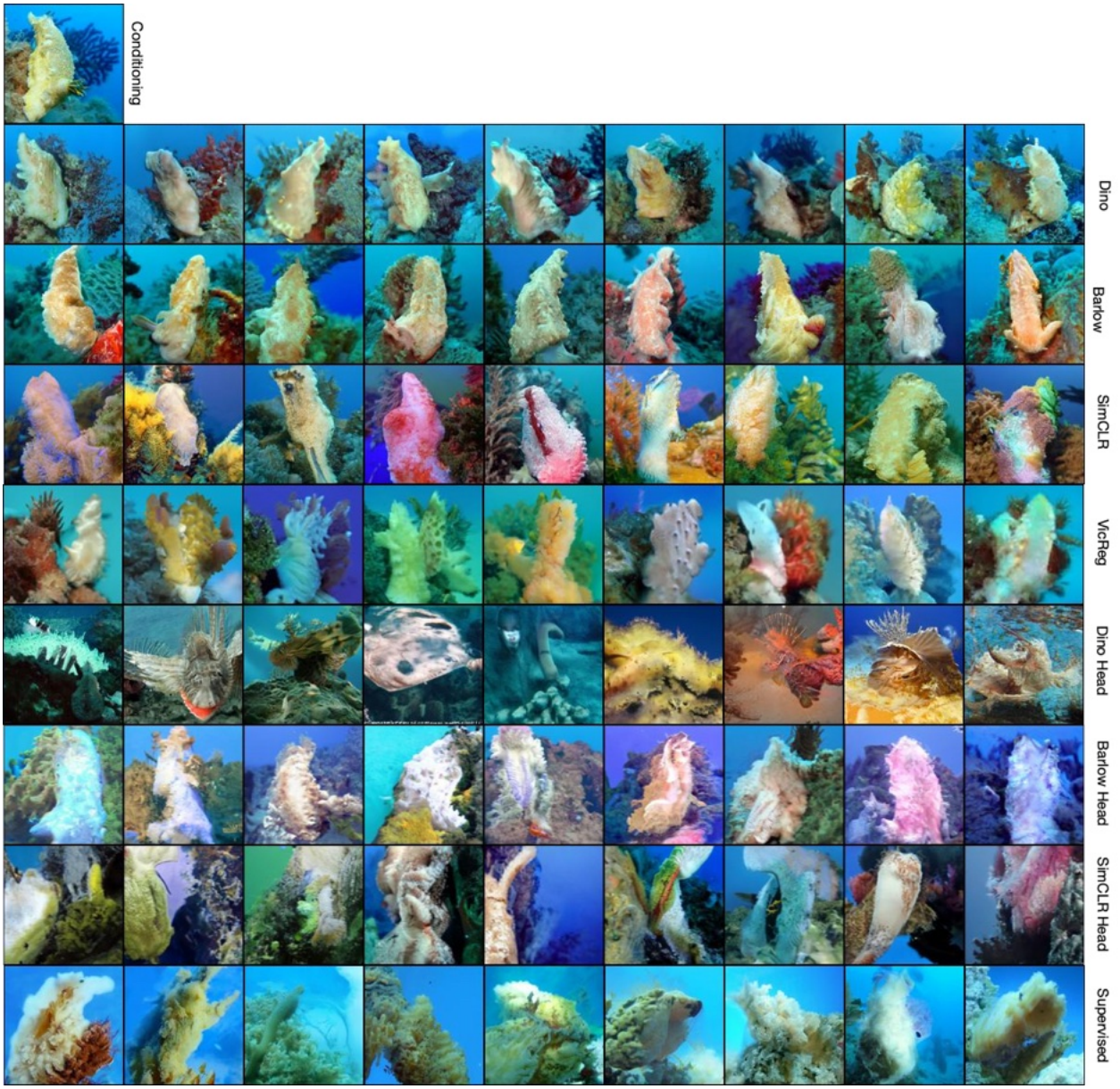


Fig. 19. Generated samples from RCDM trained with representation from various self-supervised models. We generate 9 samples for each model with different random seeds. We observe that the representation given by dino isn't very invariant while the one given by SimCLR or VicReg show much better invariance. We also show the samples of RCDM trained on the representation given by the projector (The embedding on which is usually applied the SSL criterion). There is a much higher variability in the generated samples. Maybe too much to be used for a classification task since we can observe class crossing.

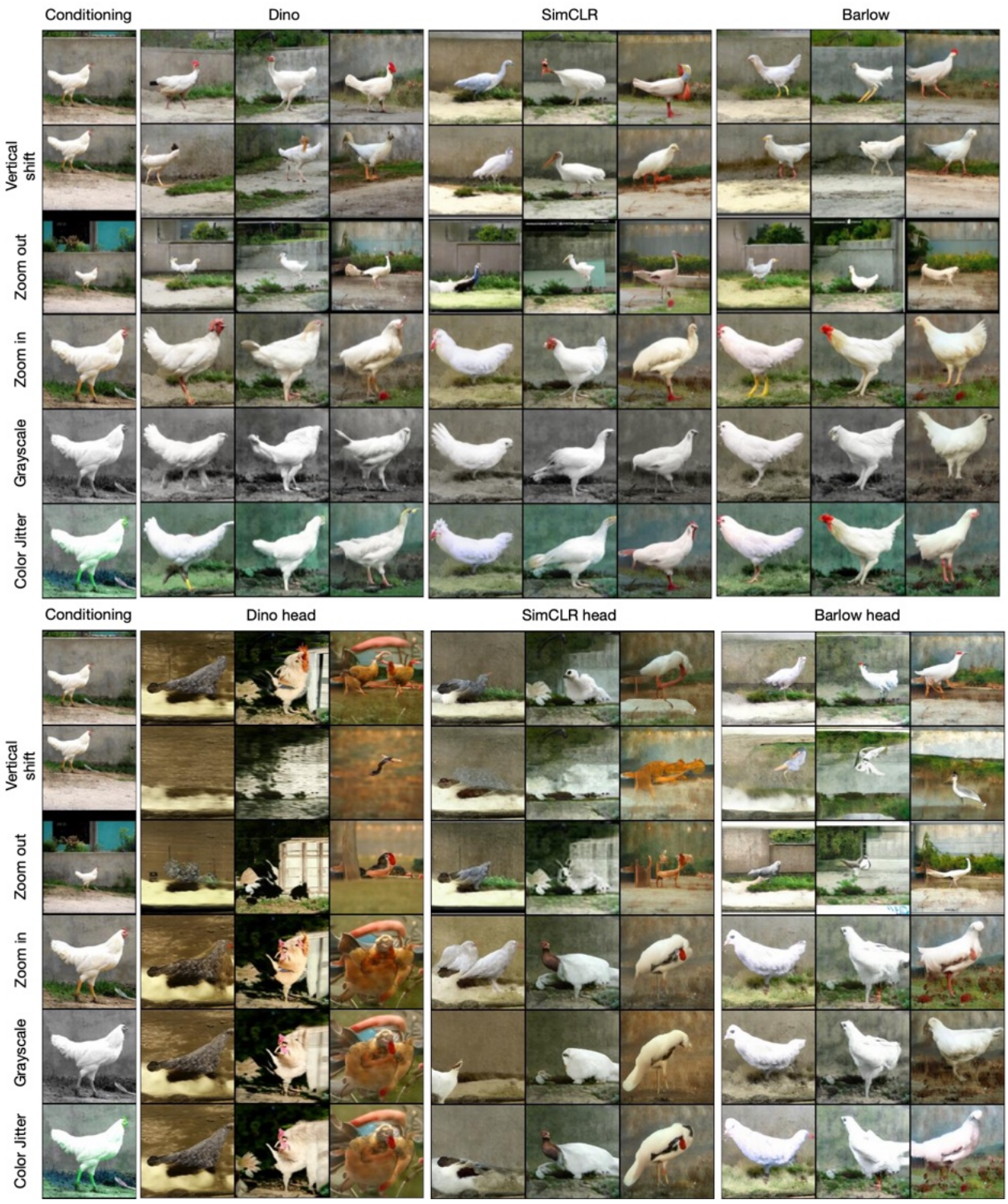


Fig. 20. We compare how much the samples generated by RCDM change depending on different transformations of a given image and the model and layer used to produces the representation. Top half uses 2048 representation. Bottom half uses the lower dimensional projector head embedding. We observe that using the projector head representation leads to a much larger variance in the generated samples whereas using the traditional backbone (2048) representation leads to samples that are very close to the original image. We also observe that the projector representation seems to encode object scale, but contrary to the 2048 representation, it seems to have gotten rid of grayscale-status and background color information.

Fig. 21. Same setup as Figure 20 except with other images as conditioning.

Fig. 22. Generated samples from RCDM using the mean representation for a specific class (golden retriever) in ImageNet for various SSL models.

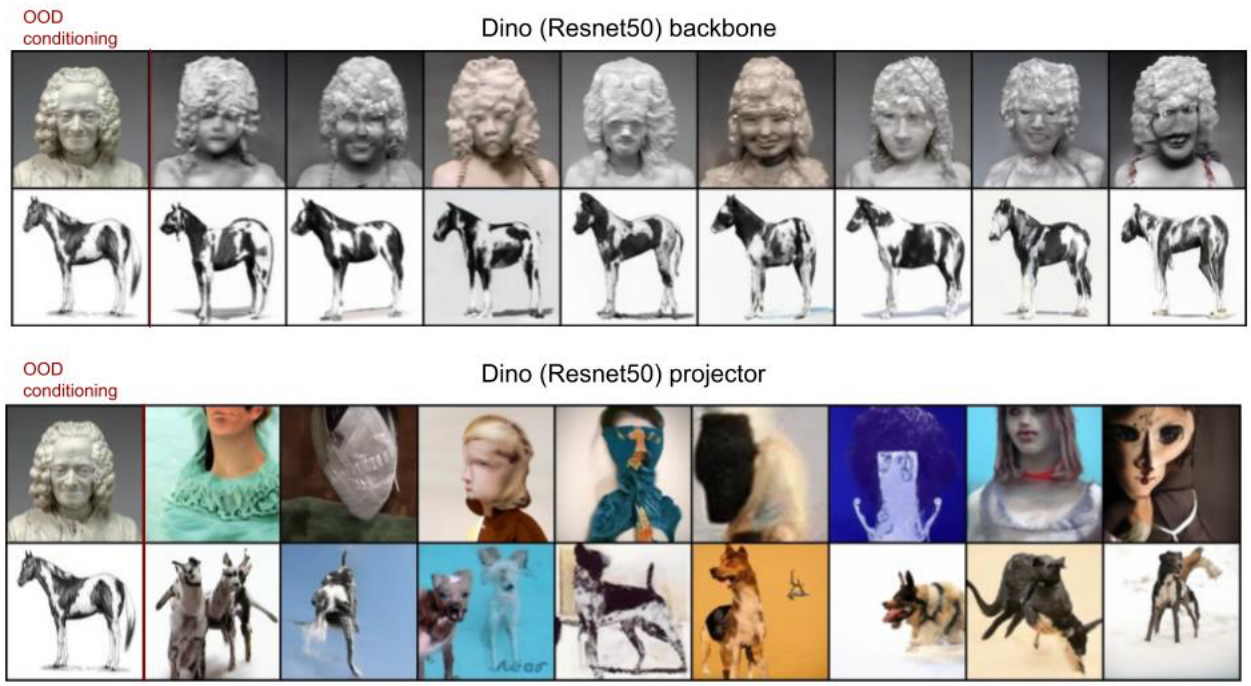


Fig. 23. We compare the visualization obtained with representations from Dino (resnet50) at the backbone level and also at the projector level on OOD images to ensure that conclusions drawn with our model about SSL representations are not specific to ImageNet. We confirm that we observe the sames phenomenons in an OOD settings as the ones we could get on an In-Distribution scenario.

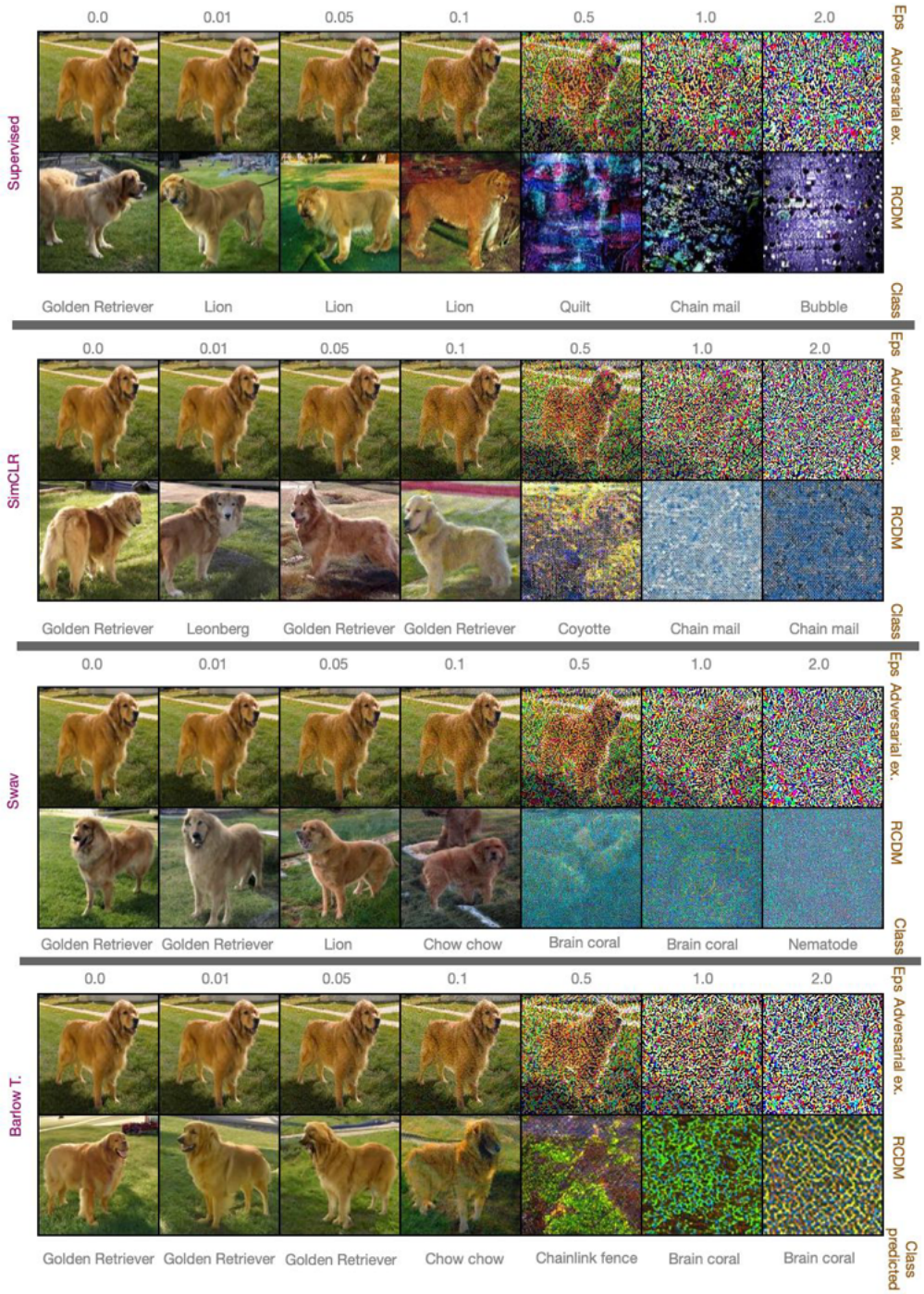


Fig. 24. Visualization of adversarial examples We use RCDM to visualize adversarial examples for different models. For each model, we trained a linear classifier on top of their representations to predict class labels for the ImageNet dataset. Then, we use FGSM attack over the trained model using a NLL loss to generate adversarial examples towards the class lion. For each model, we visualize adversarial examples for different values of ϵ which is the coefficient used in front of the gradient sign. In the supervised scenario, even for small values of epsilon which doesn't seem to change the original image, the decoded image as well as the predicted label by the linear classifier becomes a lion. However it's not the case in the self-supervised setting where the dog still get the same class or get another breed of dog as label until the adversarial attack becomes more visible to the human eye (For ϵ value superior to 0.5).

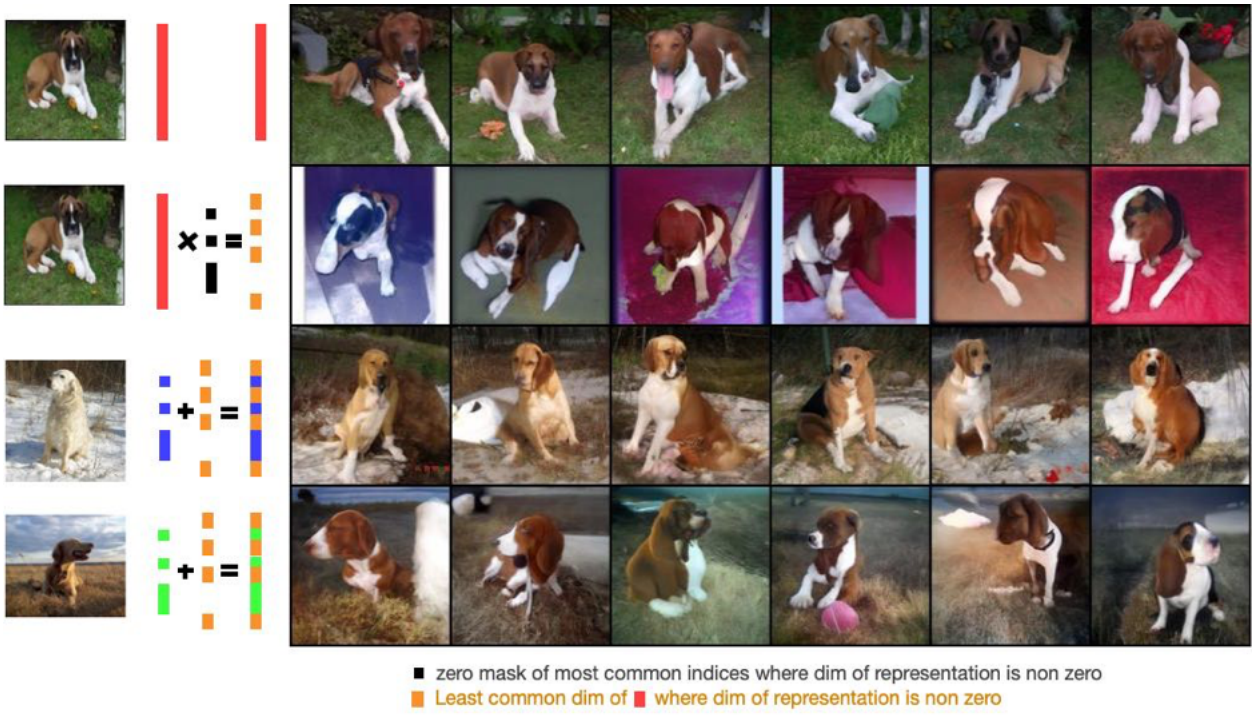


Fig. 25. Background suppression and addition Visualization of direct manipulations over the representation space. On the first row, we used the full representation of the dog's image on the top-left as conditioning for RCDM. Then, we find the most common non zero dimension across the neighborhood of the image used as conditioning. On the second row, we set these dimensions to zero and use RCDM to decode the truncated representation. We observe that RCDM produces examples of the dog with a high variety of unnatural background meaning that all information about the background is removed. In the third and forth row, instead of setting the most common non zero dimension to zero, we set them to the value of corresponding dimension of the representation associated to the image on the left. As we can see, the original dog get a new background and a new pose.

Fig. 26. Same setup as Figure 25 except that instead of using the most common non zero-dimensions as mask, we used the least common non-zero dimensions as mask. On the second row, we observe that some information about the original dog is removed such that in each column, we get a slightly different breed of dog while the background stay fixed. On the third and forth row, we saw that the information about the background (grass) is propagated through the samples (which was not the case in Figure 25).

Fig. 27. Algebraic manipulation of representations from real images (left-hand side of =) allows RCDM to generate new images with novel combination of factors. Here we use this technique with ImageNet images, to attempt background substitutions.

Fig. 28. Conditional generation with RCDM using representation extracted from a VIT-B 16 trained with Dino. This experiment shows that RCDM is able to successfully use the representation extracted form different kinds of architectures.

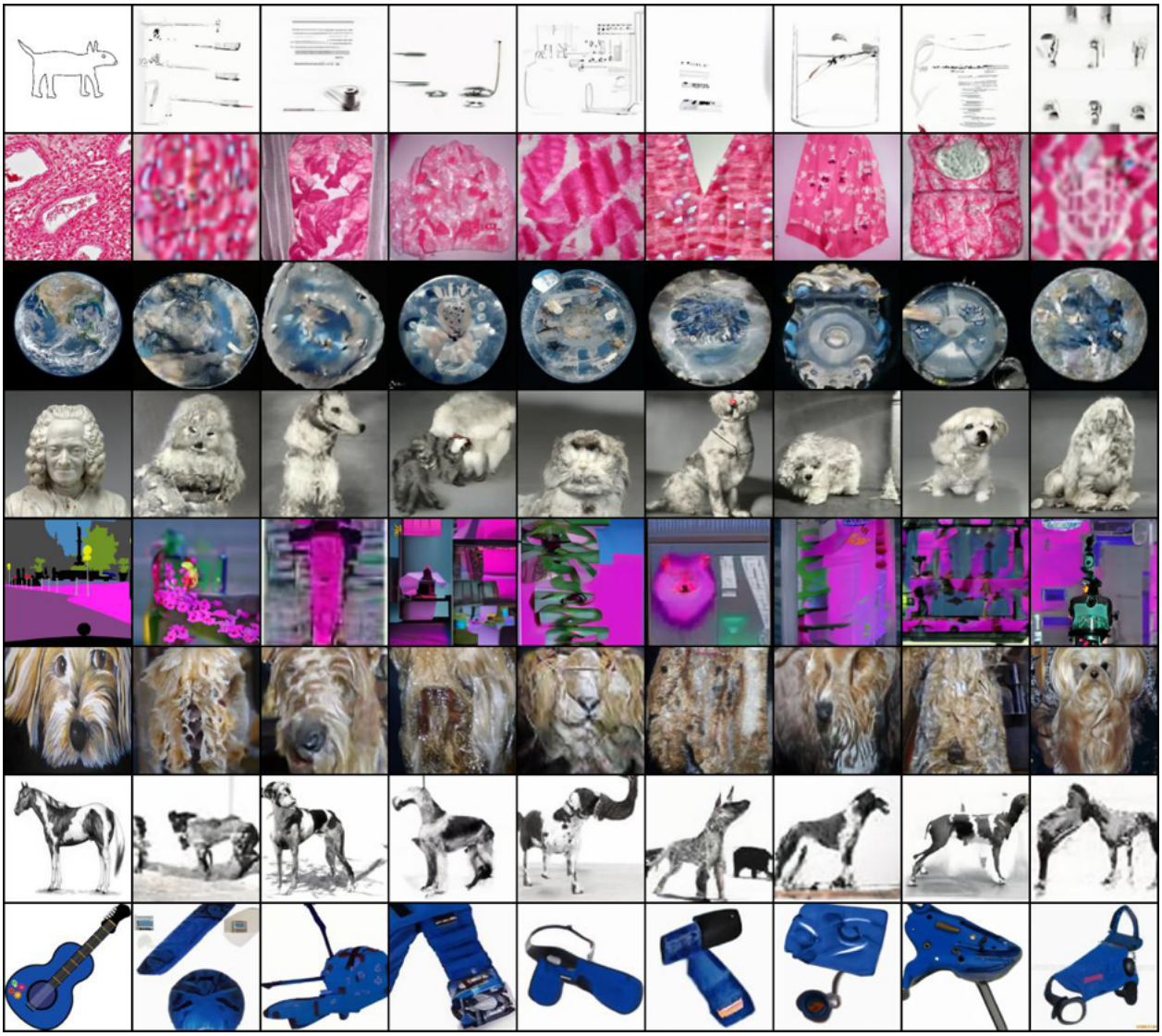


Fig. 29. Conditional generation with RCDM using representation extracted from a Resnet50 trained with VicReg using only cropping as data augmentation (thus discarding all transforms related to color change). This experiment shows that training an SSL model without learning any invariances to colors lead to learn only statistics about the colors in the representation. We can clearly see that the samples generated from the guitar are clearly following the same colors statistics as the conditioning but totally fail to reconstruct anything related to the shape information. The source for the picture of the earth is NASA.

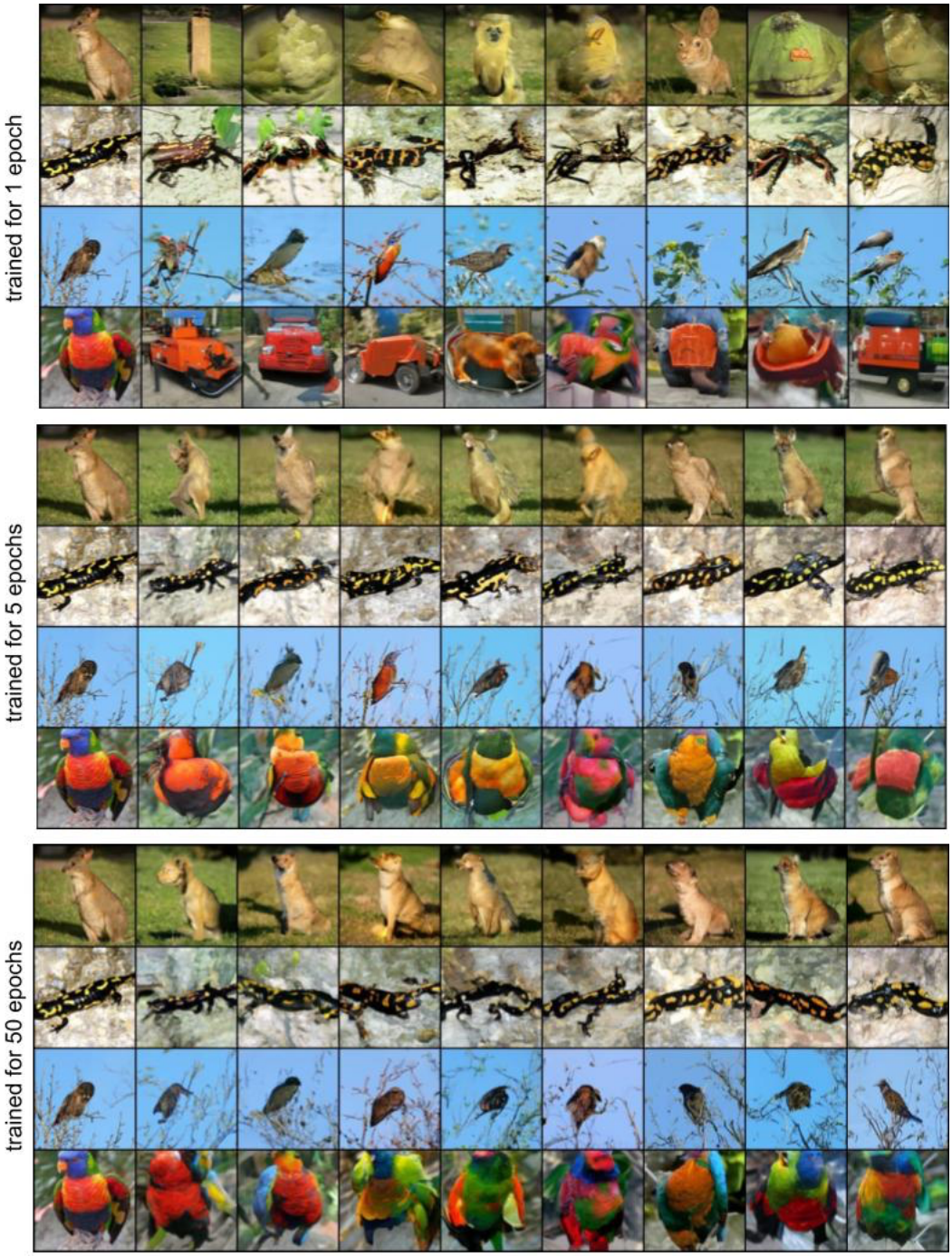
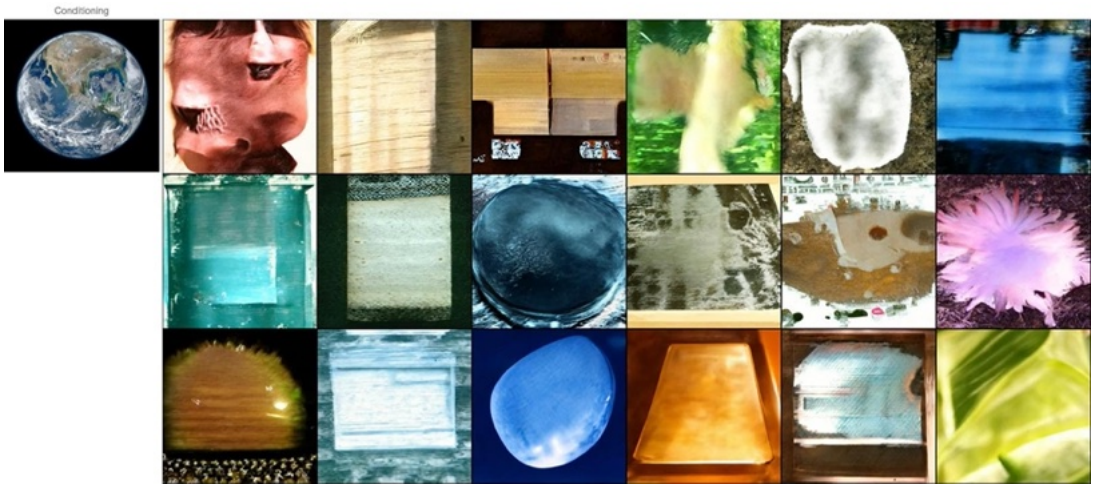
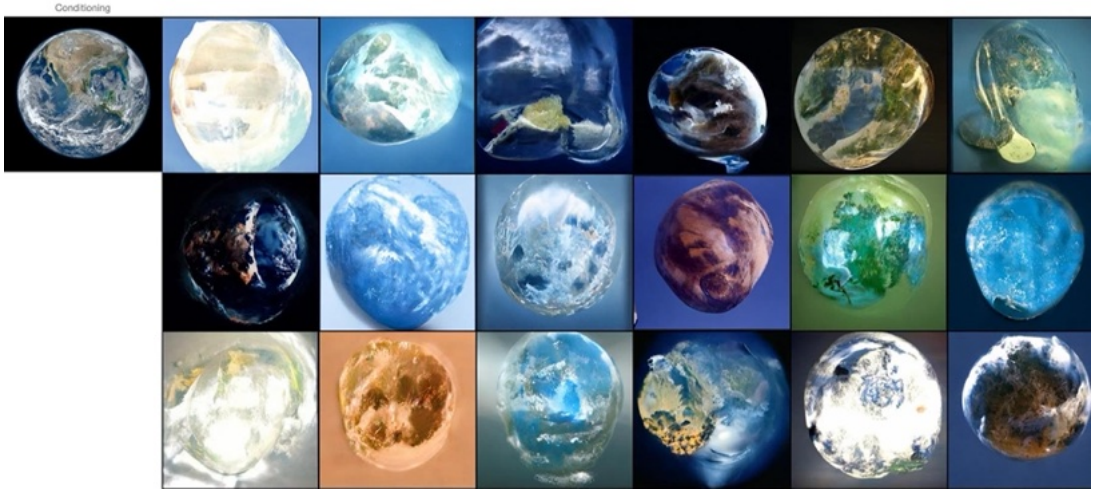


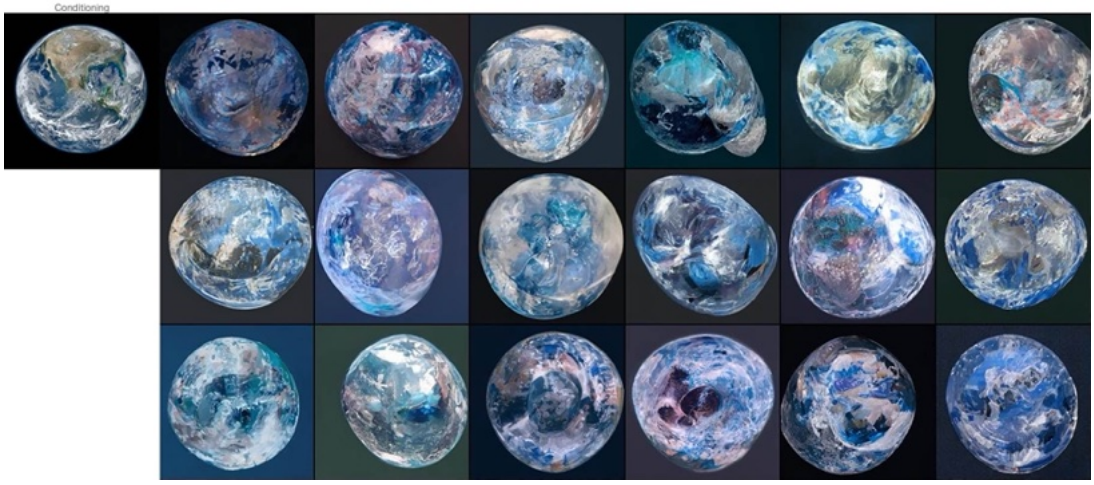
Fig. 30. Conditional generation with RCDM using representation extracted from a Resnet50 trained with VicReg for 1, 5 and 50 training epochs (a new RCDM generator is trained fully for each case). This experiment shows that the SSL model first (after 1 epoch) learns to retain mostly information about color and texture in its representation (see e.g. how conditioning on the parrot representation yields *vehicles* with similar color-themes). It encodes accurate information on the more precise shape only later in training.



(a) Earth from an untrained representation (Random initialized Resnet 50).



(b) Earth from a supervised representation (Pretrained resnet50 on ImageNet)



(c) Earth from a SSL representation (Dino Resnet50 backbone).

Fig. 31. Different samples of RCDM conditioned on a satellite image of the earth (source: NASA). We show the samples we obtained in a) when using a random initialized network to get representations, b) when using a pretrained resnet50, c) when using a self supervised model (Dino).

Appendix B

Appendix: Guillotine Regularization: Why removing layers is needed to improve generalization in Self-Supervised Learning

B.1. Datasets

In this work, we use ImageNet [56] (Term of license on https://www.image-net.org/download.php) for our experiments. We also used a synthetic 3D dataset that will be described in the next subsection.

B.1.1. 3D models dataset

We will now discuss the dataset used for figure 2. As previously mentioned, this dataset consists of 3D models from 3D Warehouse [202], freely available under a General Model License, and rendered with Blender's Python API. We alter the scene by uniformly varying the latent variables described in table 1.

Table 1. Latent variables used to generate views of 3D objects. All variables are sampled from a uniform distribution.

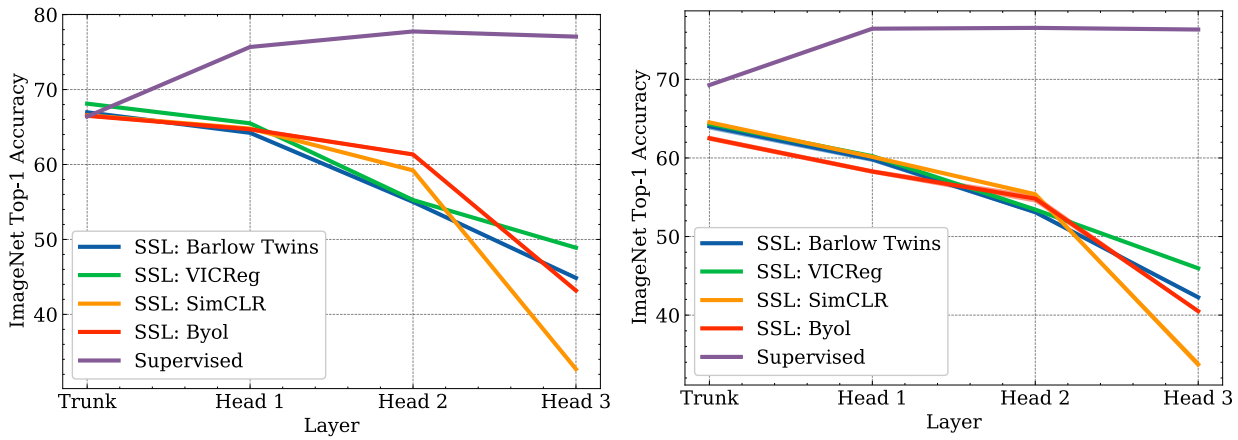
Latent variable	Min. value	Max. value
Object yaw	$-\pi/2$	$\pi/2$
Object pitch	$-\pi/2$	$\pi/2$
Object roll	$-\pi/2$	$\pi/2$
Floor hue	0	1
Spot θ	0	$\pi/4$
Spot ϕ	0	2π
Spot hue	0	1

The variety in the scenes that can be generated is illustrated in figure 1. We can see that each latent variables can significantly impact the scene, giving a significant variety in the rendered images.

Fig. 1. Rendered views of a skateboard generated by randomly sampling latent variables. The influence of each parameter is easily visible, which is expected to make their prediction easier.

B.2. Reproducibility

Our work does not introduce a novel algorithm nor a significant modification over already existing algorithm. Thus, to reproduce our results, one can simply use the public github repository of the following models: SimCLR, Barlow Twins, VicReg or the PyTorch Imagenet example (for supervised learning) with the following twist: adding a linear probe at each layer of the projector (and backbone) when evaluating the model. However, since many of these models can have different hyper-parameters, or data-augmentations, especially for the SSL models, we recommend to use a single code base with a given optimizer, a given set of data augmentations so that comparisons between models are fair and focus on the effect of Guillotine Regularization. In this paper, except if mentioned otherwise, we use as Head, a MLP with 3 layers of dimensions 2048 each (which match the number of dimensions at the trunk of a Resnet50) along with batch normalizaton and ReLU activations.



(a) Accuracy on the training set

(b) Accuracy on the validation set

Fig. 2. We measure with linear probes the accuracy at different layers on a resnet50 (as Trunk) on which we added a small 3 layers MLP (as Head) for various supervised and self-supervised methods on the training and validation set. For each method, we show the mean and standard deviation across 3 runs (The std between different runs is low). When looking at self-supervised methods, the gap in performances between the linear probe trained at different levels can be as high as 30 points of percentage.

B.3. Additional experimental results

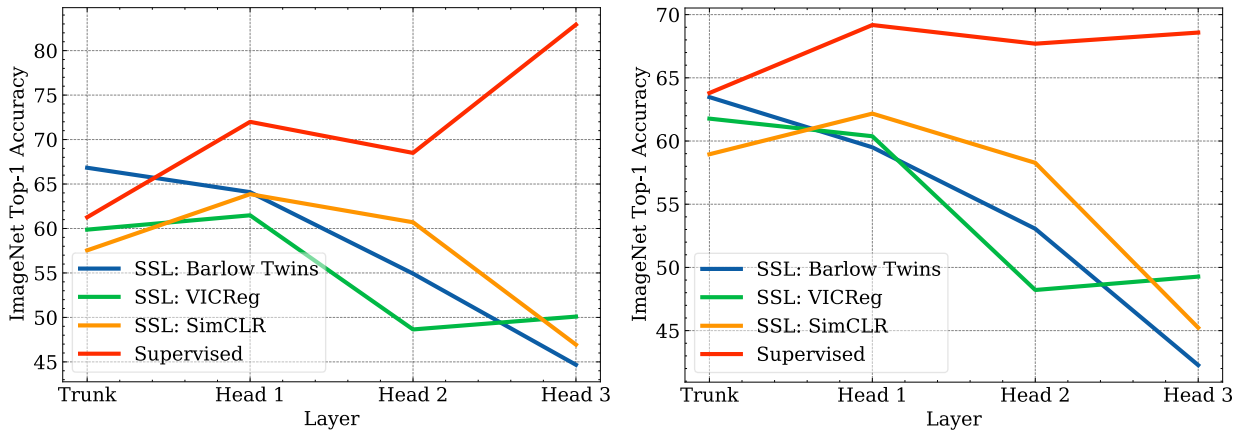
In this section, we present additional experimental results. The first one in Figure 2 is an extended version of Figure 1 with additional results on the training set. Figure 3 is a similar setup to the one in Figure 2 where we compared the performances at different layers for SSL methods and a supervised one except that we use a VIT-B instead of a Resnet50. We observe an important gap on the classification performances reached with a linear probe on different layers with the VIT-B when using SSL methods.

In Figure 4, we show how the performances at different layers change during training by using an online linear probing. At the beginning of the training the gap of performances between layers is low, however it increases significantly after 10 epochs.

In Figure 5 we show the accuracy computed with linear probes trained using projector and backbone representations. This figure is similar to Figure 6 except that we present the absolute accuracy value instead of the difference in accuracy with respect to the backbone.

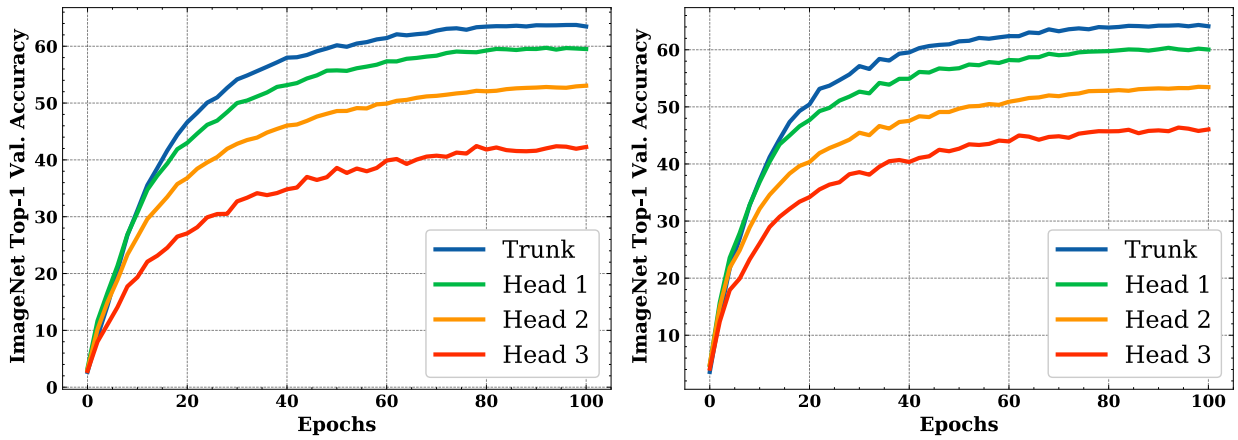
B.4. Limitations

In this work we focused mostly on analyzing the use of Guillotine Regularization in the context of Self-Supervised Learning. However, this kind of regularization might be useful for a variety of other types of training methods which we don't investigate in this paper. We also mostly focus on generalization for classification tasks, but other tasks could also been worth exploring.



(a) Accuracy on the training set **(b)** Accuracy on the validation set

Fig. 3. Same experiment as in Figure 2 but this time, we measure with linear probes the accuracy at different layers on a VIT-B (as Trunk) on which we added a small 3 layers MLP (as Head) for various supervised and self-supervised methods. Since the outputs of the VIT-B has a lower number of dimensions than a Resnet, we added at the trunk of the VIT-B a linear layer with ReLU activation to project into a 2048 dimensional vector. In the supervised learning setting, the best performances are obtained when using the last layers of the model. But, when looking at self-supervised methods, the gap in performances between the linear probe trained at different levels can be as high as 20 points of percentage. Interesting, it seems for the VIT-B that we got the best performances at Head 1 for SimCLR whereas for the ResNet, the best performances were obtained at the Trunk. It is likely that for different architectures, the optimal number of layers on which to apply Guillotine Regularization will vary.



(a) Accuracy on the training set **(b)** Accuracy on the validation set

Fig. 4. a) Accuracy of Barlow Twins through epochs computed with online linear probing at different layers. At the beginning of the training the gap in performances between the probes is small however after 10 epochs, the gap becomes larger and larger both on the training and validation set.

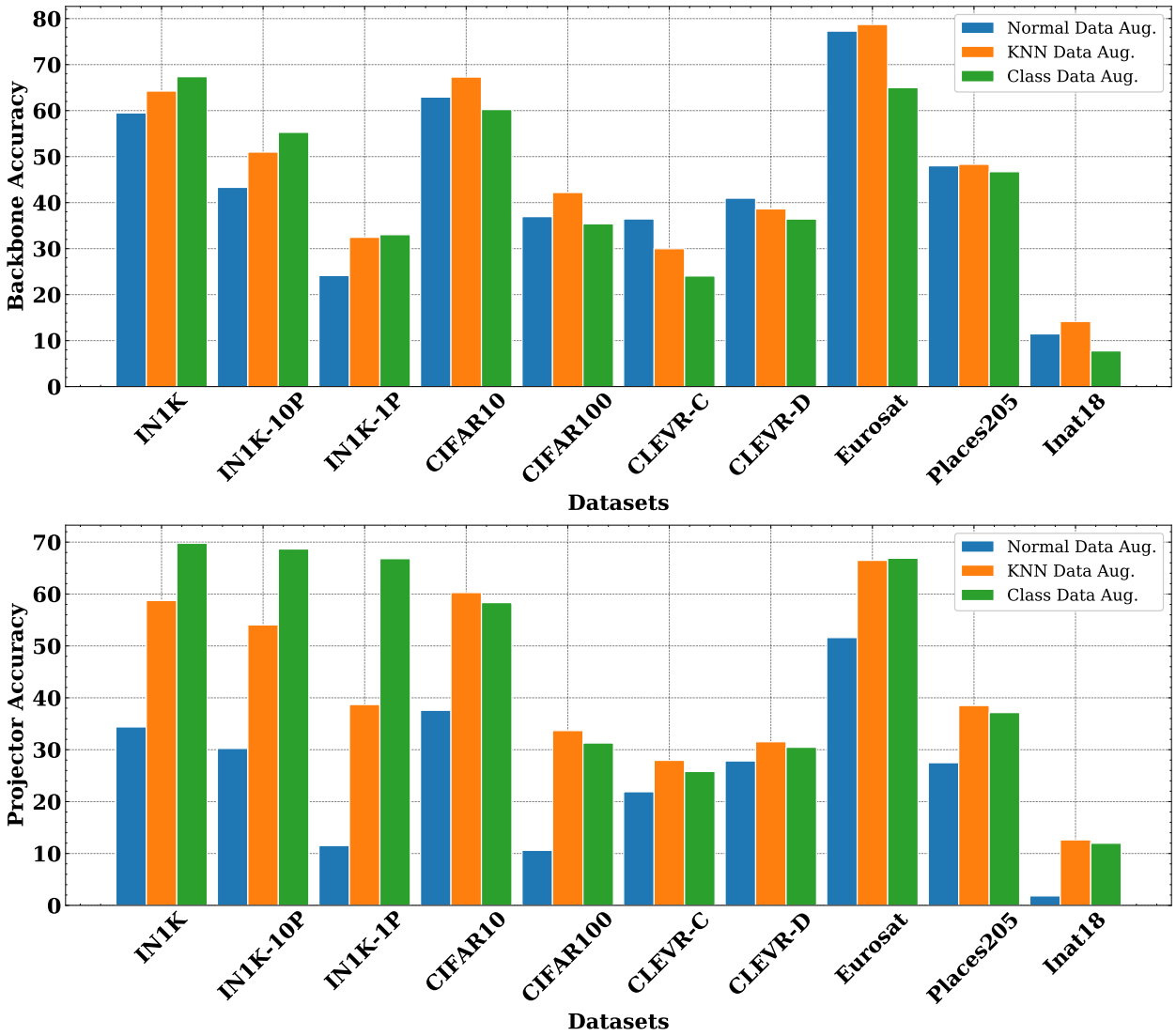
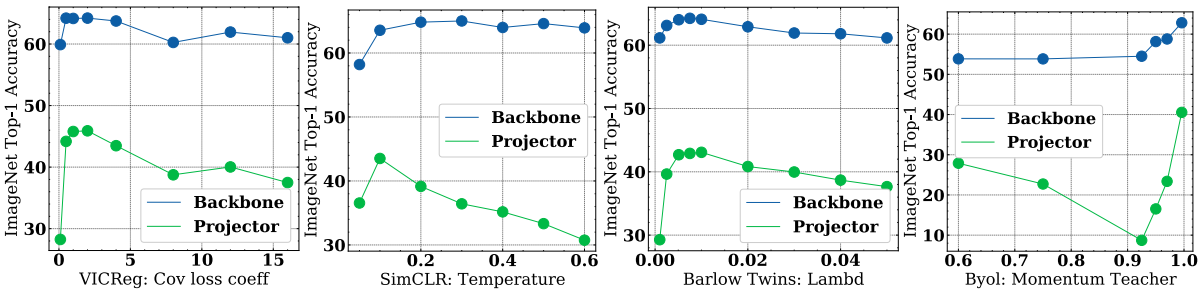
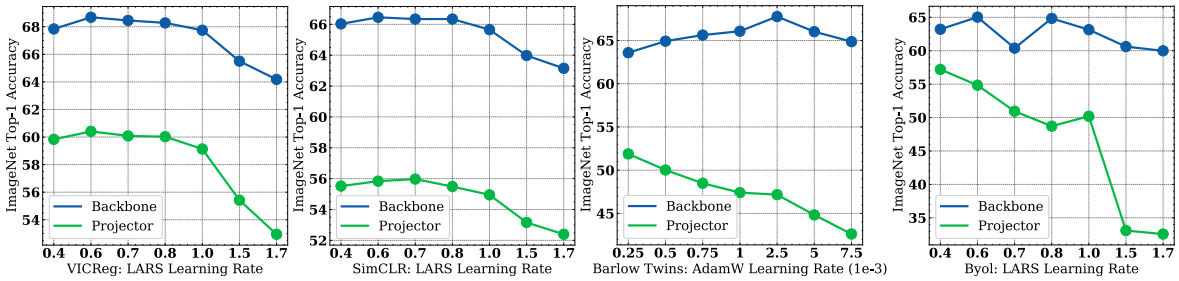


Fig. 5. Backbone and projector accuracy with linear probing with different alignment with respect to the classification downstream task. In this experiment we used SimCLR and we change how the positive pair are defined to better aligned with a classification downstream task. In blue, our baseline, we trained SimCLR with the traditional SSL data augmentations which defines the positive view as two augmentations of a same image. In orange, we use the embedding of a pretrained model to define the positive pair as two nearest neighbor under this pretrained model (while using the same data augmentation as the baseline). In green, we use a supervised class label selection to define the positive example. In this scenario, SimCLR should learn to produces similar embedding to all images belonging to a given class. All three models are trained on ImageNet (IN1K), then we evaluate them with a linear probe across a wide range of downstream tasks at the projector and backbone level. When positives pairs are defined as belonging to a given class, there is no misalignment with the imagenet classification downstream task. Thus on ImageNet-1K, ImageNet1k-10P (10% of the training set to train the linear probe) and ImageNet1k-1P (1% of the training set to train the linear probe), we observe that the performances at the projector level are much higher than the ones using the traditional SSL augmentations. Interestingly, the nearest neighbors heuristic reduces considerably the impact of Guillotine Regularization across several downstream tasks.



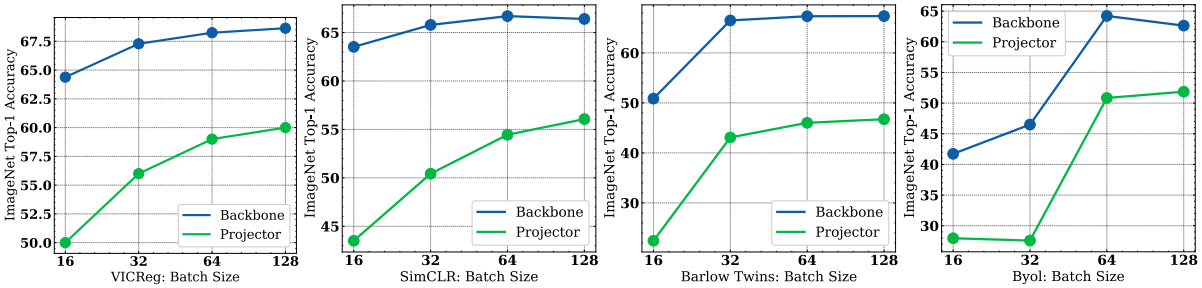
(a) VICReg **(b)** SimCLR **(c)** Barlow Twins **(d)** Byol

Figure 5.1: Loss hyper-parameters.



(e) VICReg **(f)** SimCLR **(g)** Barlow Twins **(h)** Byol

Figure 5.2: Learning rate.



(i) VICReg **(j)** SimCLR **(k)** Barlow Twins **(l)** Byol

Figure 5.3: Batch size.

Fig. 6. How different hyper-parameters impact the gap in performances between the backbone and projector representation. We train SimCLR, VicReg, Barlow Twins and Byol with different hyper-parameters and evaluate with a linear prob, the performances at the backbone but also at the projector level on ImageNet classification task. For each model, we observe that the accuracy given by the linear probe at the backbone level is fairly stable across the grid search of hyper-parameters while the linear probe at the projector level can reach very low accuracy. This highly that the probes performances at different layers might not be always correlated with each others.

Appendix C

Do SSL Models Have Déjà Vu? A Case of Unintended Memorization in Self-supervised Learning

C.1. Limitations and societal impact

Limitations. Our work sets out to define, quantify, and visualize data memorization in SSL. Our tests guide us towards potential mitigation strategies. However, note that these strategies are distinct from provable privacy (e.g. DP), and do not guarantee that data is not memorized. It is possible that — even if our tests detect no memorization — data is being memorized in some other fashion, and could be detected with a different test. Furthermore, we focus on detecting image memorization with a curated, de-duplicated dataset (ImageNet-1k), which may over- or underestimate data memorization in practice. We chose this in order to claim the learning algorithm as the cause for memorization as opposed to the dataset itself. It is possible that models exhibit different memorization behavior on larger, less curated datasets. With orders of magnitude more data it is possible that memorization is reduced, but with more data duplication it also may be exacerbated.

Societal Impact. Our work's findings have a critical societal impact from a privacy perspective. We show that it is possible for SSL—an increasingly popular learning paradigm—to memorize training images, which could have significant privacy implications. This direction of research is important if we want to understand how we can train such models without exposing user data. Additionally, our proposed mitigation strategies point to the possibility of having strong privacy without significant loss in utility. Ultimately, we open a promising direction towards making SSL vision models more secure.

C.2. Experimental details

C.2.1. Details on dataset splits

There are 1,281,167 training images in the ImageNet dataset. Within these images, only 456,567 of them have bounding boxes annotations (which are needed to compute the Deja Vu score). The private set $\mathcal{A}$ and $\mathcal{B}$ are sampled from these 456,567 bounding boxes annotated images in such a way that set $\mathcal{A}$ and $\mathcal{B}$ are disjoint.

If we remove the 456,567 bounding boxes annotated images from the 1,281,167 training images, we get 824,600 remaining images without annotations which never overlap with $\mathcal{A}$ or $\mathcal{B}$. From this set of 800K images, we took 500k images as our public set X. So now, we have three non overlapping sets $\mathcal{A}$, $\mathcal{B}$, and X. Then, if we remove the 500K public set images from the 824,600 images without annotations, it leaves us with 324,600 images that are neither in $\mathcal{A}$, $\mathcal{B}$ or X. For simplicity, let us call this set of remaining 324,600 images the set $\mathcal{C}$. Then, we have split the entire ImageNet training set into four non-overlapping splits called $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ and X.

When running our experiments with a small number of training images, we only use the set $\mathcal{A}$ to train SSL_A and the set $\mathcal{B}$ to train SSL_B and then use the set X as a public set for evaluation. However, to run larger scale experiments, we use as additional training data for SSL_A and SSL_B: the images sampled from the set $\mathcal{C}$. Here, SSL_A will still be trained on set $\mathcal{A}$ but it will be augmented with images from set $\mathcal{C}$. The same goes for SSL_B which will still be trained on the set $\mathcal{B}$ but augmented with images from the set $\mathcal{C}$. As such, some images sampled from $\mathcal{C}$ to train SSL_A or to train SSL_B might overlap. However, this is not an issue since the evaluation is done using only images from the bounding boxes annotated set $\mathcal{A}$ and set $\mathcal{B}$ which are never overlapping.

To identify memorization, our tests only attempt to infer the labels of the unique examples $\overline{\mathcal{A}}$ and $\overline{\mathcal{B}}$ that differentiate the two private sets. The periphery crop, $\text{crop}(A_i)$, is computed as the largest possible crop that does not intersect with the foreground object bounding box. In some instances the largest periphery crop is small, and not high enough resolution to get a meaningful embedding. To circumvent this, we only run the test on bounding box examples where the periphery crop is at least 100×100 pixels.

Each size of training set, 100k to 500k, includes an equal number of examples per class in both sets $\mathcal{A}$ and $\mathcal{B}$. The total bounding box annotated examples of each class are evenly divided between $\overline{\mathcal{A}}$ and $\overline{\mathcal{B}}$. The remaining examples in each class are the examples from $\mathcal{C}$. We reiterate that the bounding box examples in set $\mathcal{A}$ are *unique* to set $\mathcal{A}$, and thus can only be memorized by SSL_A.

The disjoint public set, X, contains ground truth labels but no bounding-box annotations. The size and content of X remains fixed for all tests.

C.2.2. Details on the training setup

Model Training: We use PyTorch [163] with FFCV-SSL [32]. All models are trained for 1000 epochs with model checkpoints taken at 50, 100, 250, 500, 750, and 1000 epochs. We note that 1000 epochs is used in the original papers of both VICReg and SimCLR. All sweeps of epochs use the 300k dataset. All sweeps of datasets use the final, 1000 epoch checkpoint. We use a batch size of 1024, and LARS optimizer [233] for all SSL models. All models use Resnet101 for the backbone. As seen in Appendix C.3.4, a Resnet50 backbone results in *déjà vu* consistent with that of Resnet101.

VICReg Training: VICReg is trained with the 3-layer fully connected projector used in the original paper with layer dimensions 8192-8192-8192. The invariance, variance, and covariance parameters are set to $\lambda = 25, \mu = 25, \nu = 1$, respectively, which are used in the original paper [23]. The LARS base learning rate is set to 0.2, and weight decay is set to 1e-6.

SimCLR Training: SimCLR is trained with the 2-layer fully connected projector used in the original paper with layer dimensions 2048-256. The temperature parameter is set to $\tau = 0.15$. The LARS base learning rate is set to 0.3, and weight decay is set to 1e-6.

Supervised Training: Unlike the SSL models, the supervised model is trained with label access using cross-entropy loss. To keep architectures as similar as possible, the supervised model also uses a Resnet101 backbone and the same projector as VICReg. A final batchnorm, ReLU, and linear layer is added to bring the 8192 dimension projector output to 1000-way classification activations. We use these activations as the supervised model's projector embedding. The supervised model uses the LARS optimizer with learning rate 0.2.

C.2.3. Details on the evaluation setup

KNN: For each test, we build two KNN's: one using the target model, SSL_A (or CLF_A), and one using the reference model SSL_B (or CLF_B). As depicted in Figure 2, each KNN is built using the projector embeddings of all images in the public set $\mathcal{X}$ as the neighbor set. When testing for memorization on an image $A_i \in \mathcal{A}$, we first embed $\text{crop}(A_i)$ using SSL_A, and find its $K = 100$ L_2 nearest neighbors within the SSL_A embeddings of $\mathcal{X}$. See section C.3.2 for a discussion on selection of K. We then take the majority vote of the neighbors' labels to determine the class of A_i. This entire pipleline is repeated using reference model SSL_B and its KNN to compute reference model accuracy.

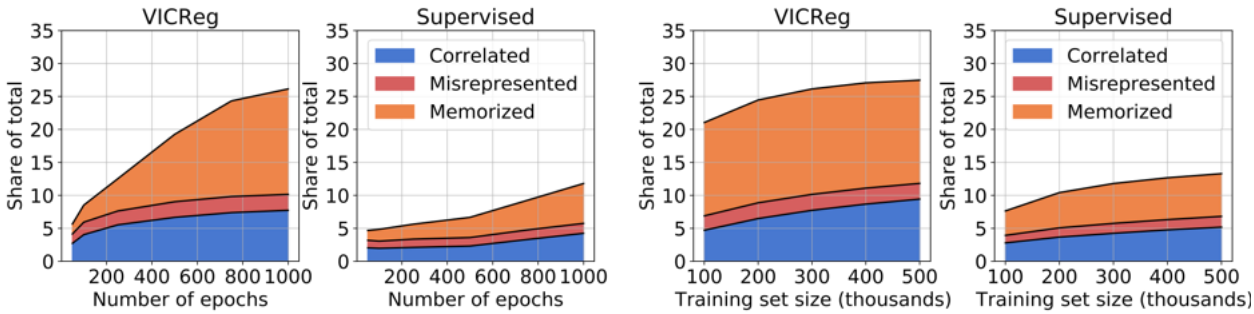
In practice, all of our quantitative tests are repeated once with SSL_A as the target model (recovering labels of images in set $\mathcal{A}$) and again with SSL_B as the target model (recovering labels of images in set $\mathcal{B}$). All results shown are the average of these two tests. Throughout the paper, we describe SSL_A as the target model and SSL_B as the reference model for ease of exposition.

RCDM: The RCDM is trained on a face-blurred version of ImageNet [57] and is used to decode the SSL backbone embedding of an image back into an approximation of the original image. All RCDMs are trained on the public set of images $\mathcal{X}$ used for the KNN. A separate RCDM must be trained for each SSL model, since each model has a unique mapping from image space to embedding space.

At inference time, the RCDM is used to reconstruct the foreground object given only the periphery cropping. To produce this reconstruction, the RCDM needs an approximation of the backbone embedding of the original image. The backbone of image A_i is approximated by **1)** computing crop embedding $\text{SSL}_A^{\text{proj}}(\text{crop}(A_i))$, **2)** finding the five public set nearest neighbors of the crop embedding, and **3)** averaging the five nearest neighbors' backbone embeddings. In practice, these public set nearest neighbors are often a very good approximation of the original image, capturing aspects like object class, position, subspecies, etc..

C.3. Additional quantitative experiments

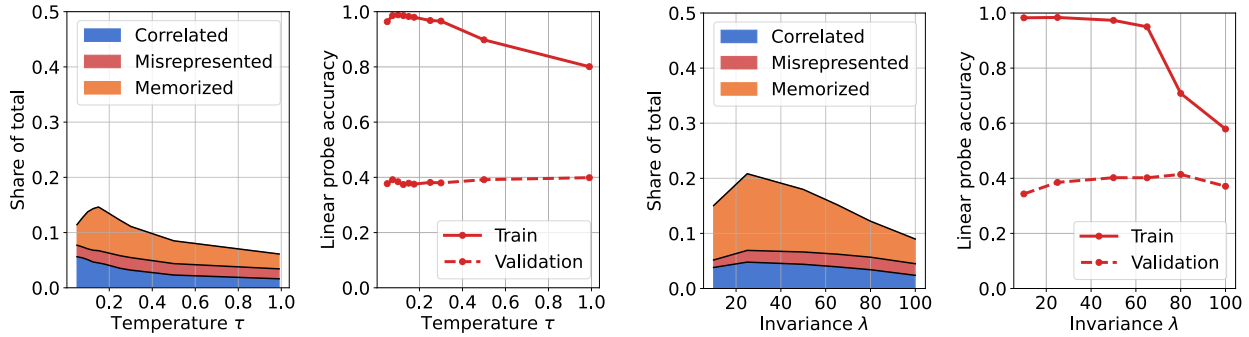
C.3.1. Sample-level memorization



(a) Categories of training samples vs. number of epochs

(b) Categories of training samples vs. training set size

Fig. 1. Partition of samples $A_i \in \mathcal{A}$ into the four categories: unassociated (not shown), memorized, misrepresented and correlated. The memorized samples—ones whose labels are predicted by KNN_A but not by KNN_B—occupy a significantly larger share for VICReg compared to the supervised model, indicating that sample-level *déjà vu* memorization is more prevalent in VICReg.



(a) SimCLR

(b) VICReg

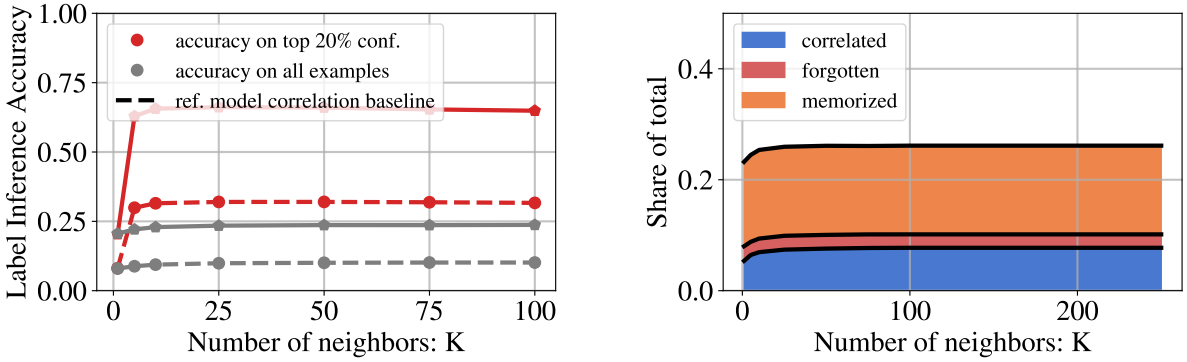
Fig. 2. Effect of SSL hyperparameter on *déjà vu* memorization. The left plot of Figures 2a and 2b show the size of the memorized set as a function of the temperature parameter for SimCLR and invariance parameter for VICReg, respectively. *Déjà vu* memorization is the highest within a narrow band of hyperparameters, and one can mitigate against *déjà vu* memorization by selecting hyperparameters outside of this band. Doing so has negligible effect on the quality of SSL embeddings as indicated by the linear probe accuracy on ImageNet validation set.

Many SSL algorithms contain hyperparameters that control how similar the embeddings of different views should be in the training objective. We show that these hyperparameters directly affect *déjà vu* memorization. Figure 2 shows the size of the memorized set for SimCLR (left) and VICReg (right) as a function of their respective hyperparameters, τ and λ. We

observe that the memorized set is largest within a relatively narrow band of hyperparameter values, indicating strong *déjà vu* memorization. By selecting hyperparameters outside this band, *déjà vu* memorization sharply decreases while the linear probe validation accuracy on ImageNet remains roughly the same.

C.3.2. Selection of K for KNN

In this section, we describe the impact of K on the KNN label inference accuracy.



(a) VICReg Accuracy **(b)** VICReg with share of memorized examples

Fig. 3. Impact of K on label inference accuracy for target and reference models. **Left:** the population-level label inference accuracy experiment of Section 4.4.1 on VICReg vs. K. **Right:** the individualized memorization test of Section 4.4.2 on VICReg vs. K. In both cases, we see that our tests are relatively robust to choice of K beyond $K = 50$.

Figure 3 above shows how the tests of Section 4.4 change with number of public set nearest neighbors K used to make label inferences. Both tests are relatively robust to any choice of K. Results are shown on VICReg trained for 1k epochs on the 300k dataset. We see that any choice of K greater than 50 and less than the number of examples per class (300, in this case) appears to have good performance. Since our smallest dataset has 100 images per class, we chose to set $K = 100$ for all experiments.

C.3.3. Effect of SSL criteria

We repeat the quantitative memorization tests of Section 4.4 on different models: VICReg[23], Barlow-Twins[236], Dino[42], Byol[93], SimCLR[238] and a supervised model in figure 4. We observe differences between SSL training criteria with respect to Dejavu memorization. The easy ones to attack are VICReg and Barlow Twins whereas SimCLR and Byol are more robust to these attacks. While the degree of memorization appears to be reduced for SimCLR compared with VICReg, it is still stronger than the supervised baseline.

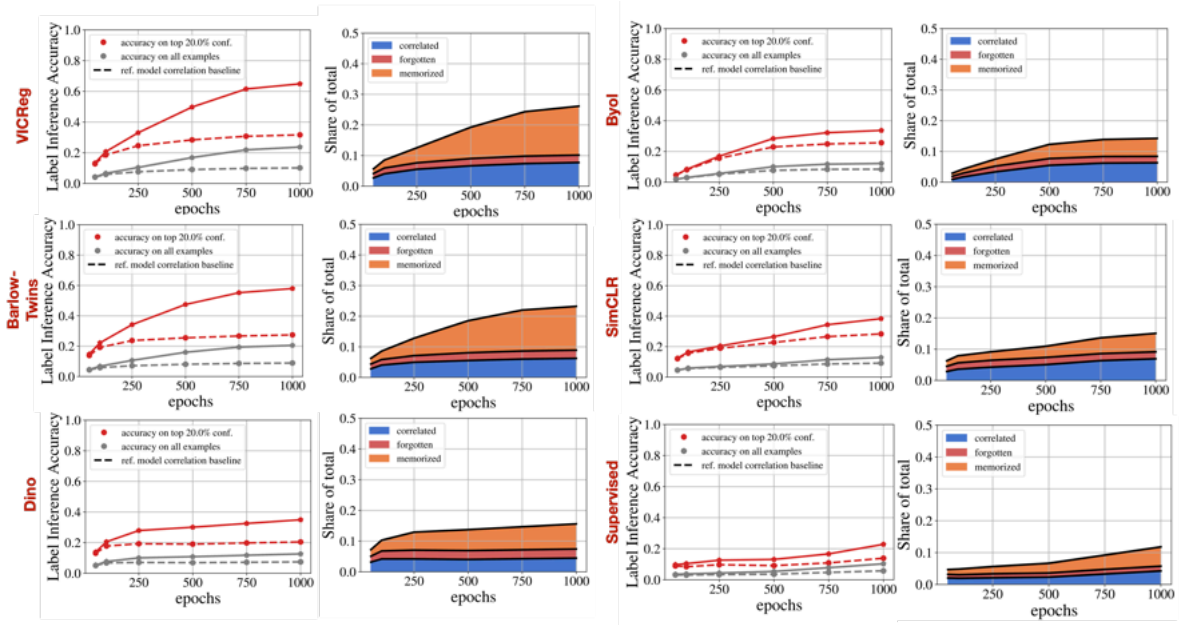


Fig. 4. Comparison of *déjà vu* memorization for VICReg, Barlow Twins, Dino, Byol, SimCLR, and a supervised model. All tests are described in Section 4.4. We are showing *déjà vu* vs. number of training epochs. We see that SimCLR (center row) shows less *déjà vu* than VICReg, yet marginally more than the supervised model. Even with this reduced degree of memorization, we are able to produce detailed reconstructions of training set images, as shown in Figures 6 and 9.

C.3.4. Effect of Model Architecture and Complexity

Results shown in the main paper use Resnet101 for the model backbone. To understand the relationship between *déjà vu* and overparameterization, we compare with the smaller Resnet50 and Resnet18 in Figure 5. Overall, we find that increasing the number of parameters of the model leads to higher degree of *déjà vu* memorization. The same trend holds when using Vision Transformers (VIT-Tiny, -Small, -Base, and -Large with patch size of 16) of various sizes as the SSL backbone, instead of a Resnet. This highlights that *déjà vu* memorization is not unique to convolution architectures.

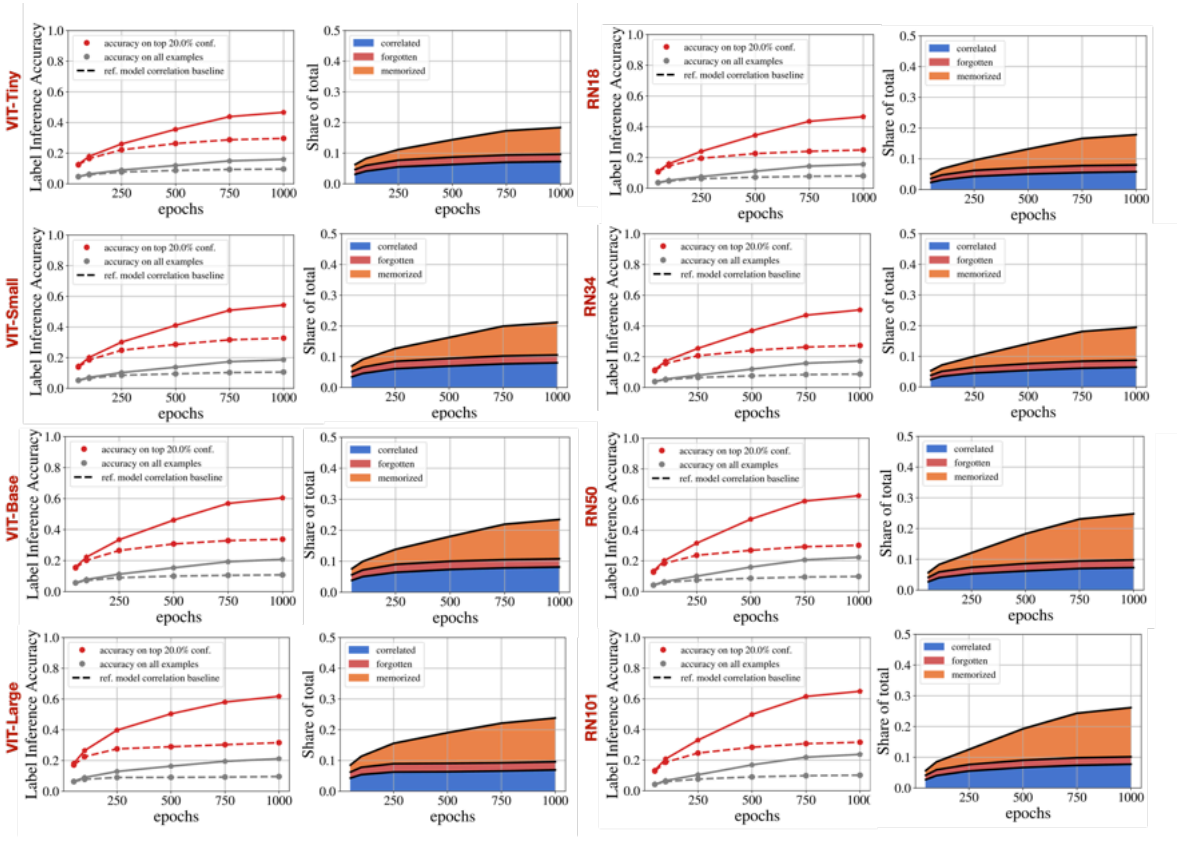


Fig. 5. Comparison of VICReg *déjà vu* memorization for different architectures and model sizes. On the left, we present deja vu memorization using VIT architectures (from vit-tiny in the first row to vit-base in the last row). On the right, we use Resnet based architectures (from resnet18 in the first row to resnet101 in the last row). All tests are described in Section 4.4, with the plots showing *déjà vu* vs. number of training epochs. Reducing model complexity from Resnet101 to Resnet18 or from Vit-Large to Vit-tiny has a significant impact on the degree of memorization.

C.3.5. The impact of finetuning

Self-Supervised learning is often used as a pre-taining strategy. In many instances, practitioners are using frozen features extracted from the model to train a simple KNN or linear head to learn to solve a downstream task. This is why we have mostly focused this work on using a set of frozen features from the pre-trained model. However, another common strategy is to fine-tune the model to solve the downstream task. In Figure 6, we show how the Dejavu score changes when fine-tuning a pretrained VICReg model. This pretrained model was trained on the set $\mathcal{A}$ for 1000 epochs and fine-tuned on a classification task on the set $\mathcal{A}$ for 20 epochs (which can be seen in the x axis on the figure). Interestingly, the DejaVu score decreases significantly in the first finetuning epochs while the validation accuracy is increasing. However after 5 epochs, the DejaVu score is increasing and after 20 epochs become almost as high at the original value before fine-tuning. This behavior indicates that even fine-tuning might not help in reducing DejaVu memorization.

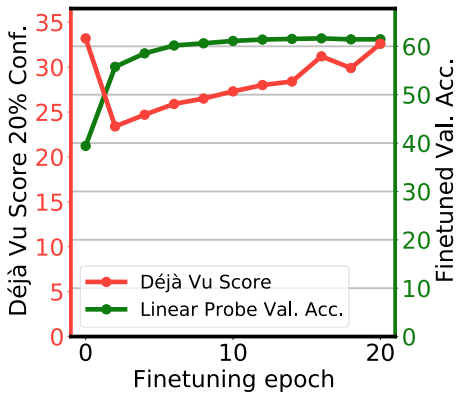
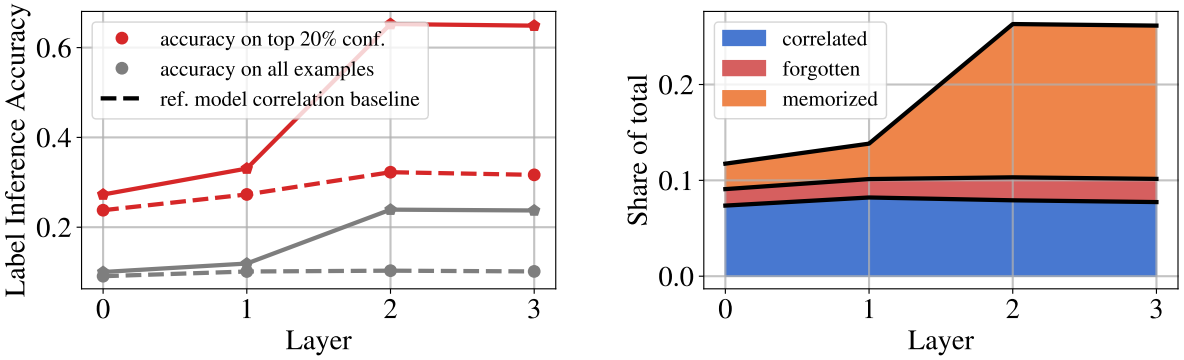


Fig. 6. DejaVu score when fine-tuning a pretrained VICReg model for 20 epochs. We observe that by fine-tuning we significantly increase the classification performances however the DejaVu score is as high as before after finetuning.

C.3.6. The impact of Guillotine Regularization on Deja Vu

In our experiments, we show *déjà vu* using the projector representation. The SSL loss directly pushes the projector representation to be invariant to random crops of a particular image. As such, we expect the projector to be the *most* overfit and produce the strongest *déjà vu* . Here, we study whether earlier representations between the projector and backbone exhibit less *déjà vu* memorization. This phenomenon – 'guillotine regularization' – has recently been studied from the perspective of generalization in Bordes et al. [30]. Here, we study it from the perspective of *memorization*.

To show how guillotine regularization impacts *déjà vu* , we repeat the tests of Section 4.4 on each layer of the VICReg projector: the 2048-dimension backbone (layer 0) up to the projector output (layer 3). We evaluate whether memorization is indeed reduced for the more *regularized* layers between the projector output and the backbone.



(a) population level accuracy

(b) share of memorized examples

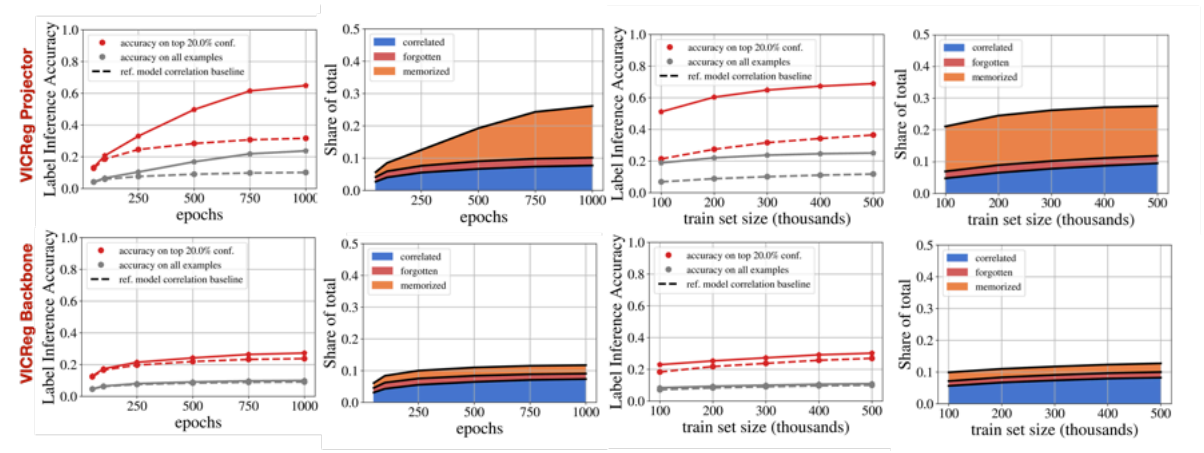
Fig. 7. *déjà vu* memorization versus layer from backbone (0) to projector output (3). The memorization tests of Section 4.4 are evaluated at each level of the VICReg projector. We see that *déjà vu* is significantly stronger closer to the projector output and nearly zero near the backbone. Interestingly, most memorization appears to occur in the final two layers of VICReg.

Figure 7 shows how guillotine regularization significantly reduces the degree of memorization in VICReg. The vast majority of VICReg's *déjà vu* appears to occur in the final two layers of the projector (2,3): in earlier layers (0,1), the label inference accuracy of the target model and reference model are comparable. This suggests that – like the hyperparameter selection of Section 4.7 – guillotine regularization can also significantly mitigate *déjà vu* memorization. In the following, we extend this result to SimCLR and

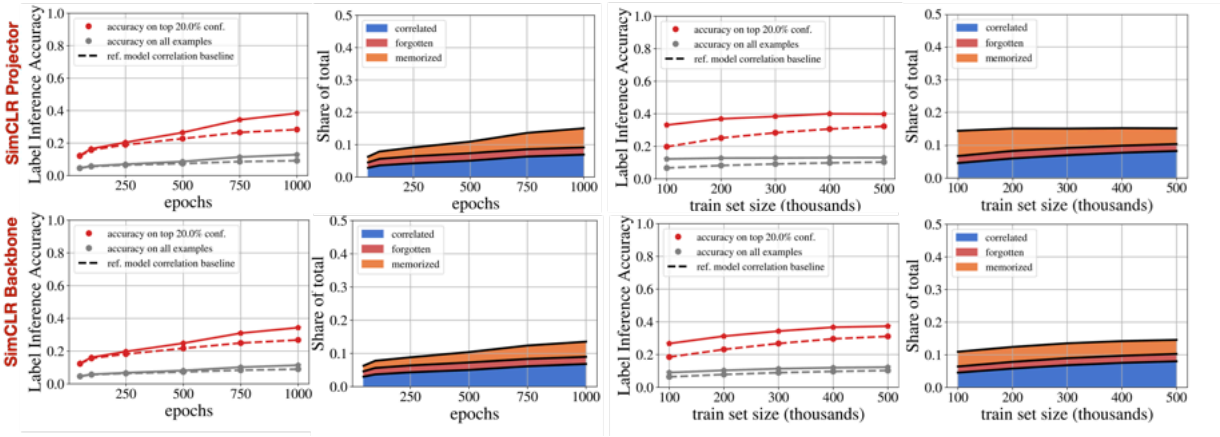
supervised models by measuring the degree of *déjà vu* in the backbone (layer 0) versus training epochs and dataset size.

Comparison of *déjà vu* in projector and backbone vs. epochs and dataset size Since the backbone is mostly used at inference time, we now evaluate how much *déjà vu* exists in the backbone representation for VICReg and SimCLR. We repeat the tests of Section 4.4 versus training epochs and train set size.

Figure 8 shows that, indeed, *déjà vu* is significantly reduced in the backbone representation. For SimCLR, however, we see that backbone memorization is comparable with projector memorization. In light of the Guillotine regularization results above, this makes some sense since SimCLR uses fewer layers in its projector. Given that we were able to generate accurate reconstructions with the SimCLR projector (see Figures 9 and 6), we now evaluate whether we can produce accurate reconstructions of training examples using the SimCLR backbone alone.



(a) VICReg

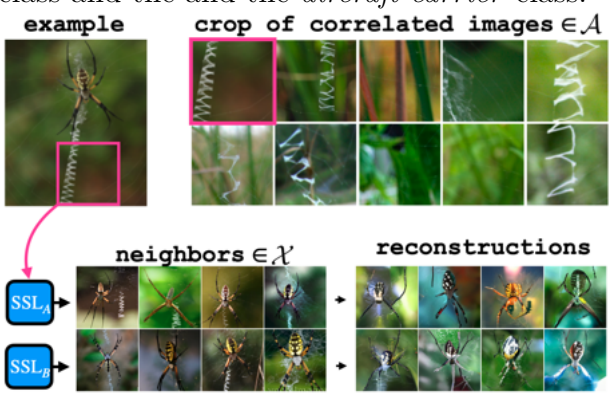


(b) SimCLR

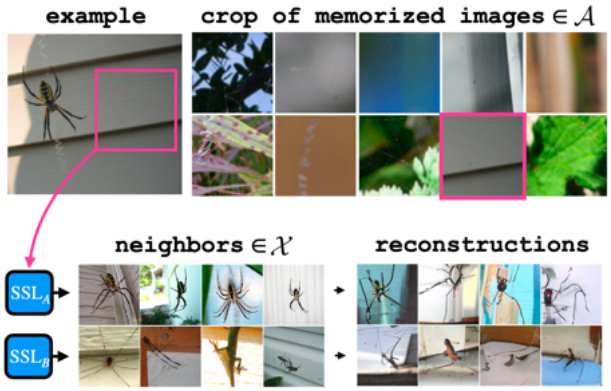
Fig. 8. Accuracy of label inference on VICReg and SimCLR using projector and backbone representations. **First two columns:** Effect of training epochs on memorization for each representation. **Last two columns:** Effect of training set size on memorization for each representation. In contrast with VICReg, the *déjà vu* memorization detected in SimCLR's projector and backbone representations is quite similar. While SimCLR's projector memorization appears weaker than that of VICReg, its backbone memorization is markedly stronger. This kind be easily explained as a byproduct of Guillotine Regularization [30], i.e. removing layers close to the objective reduce the bias of the network with respect to the training task. Since SimCLR's projector has fewer layers than VICReg's, the impact of Guillotine Regularization is less salient.

C.4. Additional reconstruction examples

The two reconstruction experiments of Section 4.5 are each exemplified within one class. However, we see strong reconstructions using SSL_A in several classes, and similar experimental results. To demonstrate this, we repeat the experiments 4.5 using the *yellow garden spider* class and the and the *aircraft carrier* class.



(a) SimCLR correlated *yellow garden spider* examples



(b) SimCLR memorized *yellow garden spider* examples

Fig. 9. Visualizing the distinction between *déjà vu* memorization and correlation in the *yellow garden spider* class. Left, we see the periphery crops of the ten 'most correlated' images: those where both KNN_A and KNN_B have high confidence. Seven of these crops clearly depict a stabilimentum: the signature zig-zag web pattern sewn by spiders of the *argiope* genus, thus revealing the concealed spider by correlation. Right, we see the periphery crops of the ten 'most memorized' images: those that have the highest confidence discrepancy between KNN_A (high confidence) and KNN_B (low confidence). Nearly all of these crops show generic blurred views of the background with no evidence of the foreground spider. Below, we show the public set nearest neighbors of the pink highlighted crop, and the RCDM reconstruction of the foreground object. We see that the target model (SSL_A) can be used to reconstruct the yellow garden spider spider in both the memorized and correlated cases. The reference model (SSL_B) can only be used to reconstruct this type of spider in the correlated case.

Selection of Memorized and Correlated Images: The images of Figure 6 and 9 were chosen methodically as follows.

Image selection: The 20 images of Figures 6 and 9 are selected deterministically using label inference accuracy and KNN confidence score. The 10 most correlated images are those images in the correlated set (both models infer label correctly) of $\mathcal{A}$ with the highest confidence agreement between models SSL_A and SSL_B. To measure confidence agreement we take the minimum confidence of the two models. The 10 most memorized images are those images in the memorized set (only target model infers the label correctly) of $\mathcal{A}$ with the highest confidence difference between models SSL_A and SSL_B.

Class selection: To find classes with a high degree of *déjà vu* , classes were sorted by the label inference accuracy gap between the target and reference model. We selected the class based on a handful of criteria. First, we prioritized classes without images of human faces, thereby removing classes like 'basketball', 'bobsled', 'train station', and even 'tench' which is a fish often depicted in the hands of a fisherman. Second, we prioritized classes that include at least ten images with a high confidence difference between the target and reference models ('most memorized' images described above) and at least ten images with high confidence agreement ('most correlated' images described above). This led us to the *dam* and *yellow garden spider* classes.

(a) extracting side of ship from VICReg **(b)** extracting front of ship from VICReg

Fig. 10. Visualization of *déjà vu* memorization beyond class label. Both SSL_A and SSL_B are VICReg models. The four images shown belong to the memorized set of SSL_A from the *aircraft carrier* class. RCDM reconstruction using embeddings from SSL_A can reveal not only the correct class label, but also the orientation of the ship: the side of the ship (left) and the front of the ship (right) given only a generic crop of the background sky and/or water. Such information does not appear to be memorized by the reference model SSL_B.

Selection of Beyond-Label-Inference Images: The images of Figure 7 and 10 were chosen methodically as follows.

Image selection: The four images of Figures 7 and 10 are selected using KNN confidence score, and, necessarily, hand picked selection for unlabeled features. Within a given class, we look at the top 40 images with highest target model KNN confidence scores. We then filter through these images to identify a distinguishable feature like different species within the same class or different object positions within the same class. This step is necessary because we are looking for features that are not labeled by ImageNet. We then choose two of these top 40 with one feature (e.g. American badger) and two with the alternative feature (e.g. European badger).

Class selection: To find classes with a high degree of *déjà vu* , classes were sorted by the target model's top-40 KNN confidence values within each class. As in the memorization vs. correlation experiment, we prioritized classes without images of human faces.

C.4.1. Reconstructions using SimCLR Backbone

The label inference results in Appendix C.3.6 show that the SimCLR backbone exhibits a similar degree of *déjà vu* memorization as the projector does. To evaluate the risk of such memorization, we repeat the reconstruction experiment of Section 4.5 on the *dam* class using the SimCLR backbone instead of its projector.

(a) First memorized dam example

(b) Second memorized dam example

Fig. 11. Instances of *déjà vu* memorization by the SimCLR backbone representation. Here, the backbone embedding of the crop is used instead of the projector embedding on the same training images used in Figure 6. Interestingly, we see that *déjà vu* memorization is still present in the SimCLR backbone representation. Here, the nearest neighbor set recovers dam given an uninformative crop of still or running water. Even without projector access, we are able to reconstruct images in set $\mathcal{A}$ using SSL_A, and are unable using SSL_B.

Figure 11 demonstrates that we are able to reconstruct training set images using the SimCLR backbone alone. This indicates that *déjà vu* memorization can be leveraged to make detailed inferences about training set images without *any* access to the projector. As such, withholding the projector for model release may not be a strong enough mitigation against *déjà vu* memorization.

C.5. Detecting *Déjà vu* without Bounding Box Annotations

The memorization tests presented critically depend on bounding box annotations in order to separate the foreground object from the periphery crop. Since such annotations are often not available, we propose a heuristic test that simply uses the lower left corner of an image as a surrogate for the periphery crop. Since foreground objects tend to be near the center of the image, the corner crop usually excludes the foreground object and does not require a bounding box annotation.

Figure 12 demonstrates that this heuristic test can successfully capture the trends of the original tests (seen in Figure 4) *without* access to bounding box annotations. However, as compared to Figure 4, the heuristic tends to slightly underestimate the degree of memorization. This is likely due to the fact that some corner crops partially include the foreground object, thus enabling the KNN to successfully recover the label with the reference model where it would have failed with a proper periphery crop that excludes the foreground object.

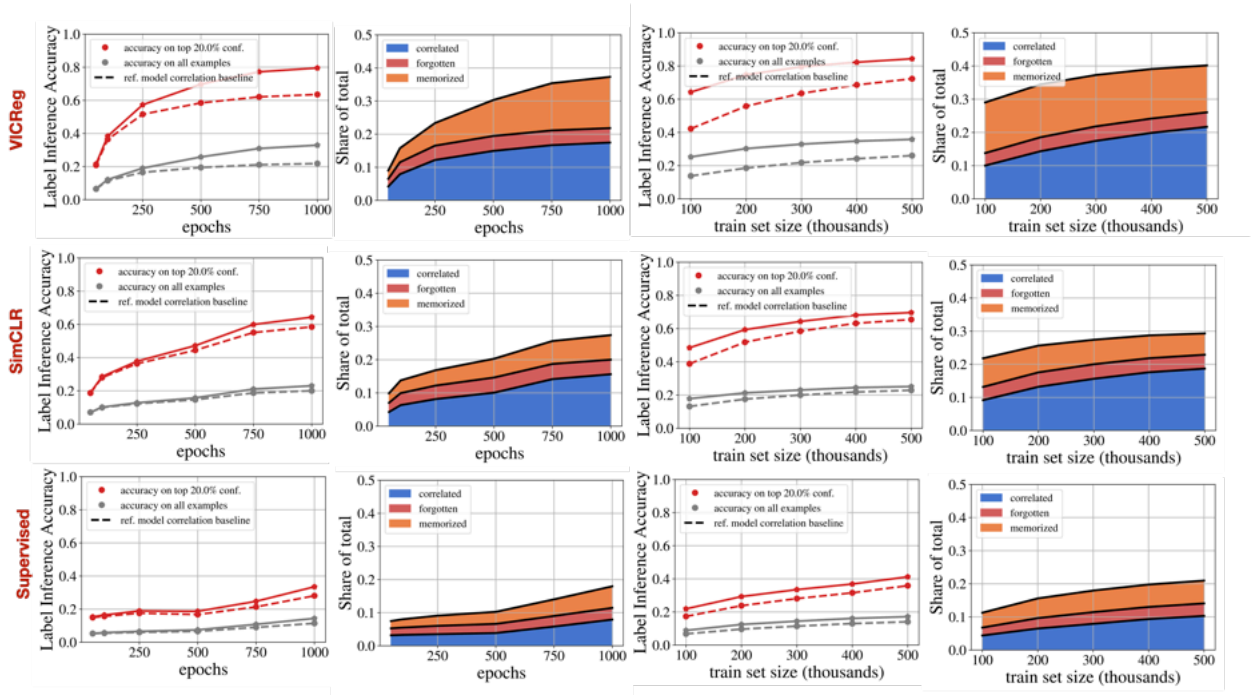


Fig. 12. Deja Vu Memorization using a simple corner crop instead of the periphery crop extracted using bounding box annotations. While the heuristic overall underestimates the degree of *déjà vu* , it roughly follows the same trends versus dataset size and training epochs. This is crucial, since it allows us to estimate *déjà vu* without access to bounding box annotations.

Appendix D

Appendix: PUG: Photorealistic and Semantically Controllable Synthetic Data for Representation Learning

D.1. Limitations and Future Work

D.1.1. Limitations

In this work, we introduced 4 new datasets that were created using the Unreal Engine. We presented a set of case studies, demonstrating how these datasets can be leveraged to improve both evaluation and training for representation learning. Deeper work would be needed to fully unlock the potential these datasets can offer. In addition, we merely scratch the surface of what a powerful engine such as the Unreal Engine can offer. With advanced techniques such as Lumen, Nanite and Megascans, it is now possible to create even more realistic environments. In addition, the datasets we provide have a single simple label, whereas future uses could easily provide detailed rich labels for the entire scene underlying each generate image.

D.1.2. Future Work

A long vision for the PUG family would be to yield a series of benchmarks that can probe the robustness of vision models. PUG: ImageNet is a first step in this direction, however we might want to get more factors of variation such as weather and occlusion. A second take would be to increase the richness of the labelling by making available detailed segmentation masks and labels. Making a short video dataset in which we have complete control over the factors is also a promising future direction since AI research on video is still far behind what can be done with images. The active learning pipeline direction is also worth to explore

since the model could bias the PUG environment to produces samples that are the best for a specific downstream task[153].

D.2. PUG Datasets

The datasets PUG: Animals, PUG: ImageNet, PUG: SPAR and PUG:AR4T are available under the **cc-by-nc license with the restrictions that they should not be use to train generative AI models**. They are available to download on the following website: `https://pug.metademolab.com/`. The datasets can be read by a torchvision ImageFolder. We have one class by folder and all the images associated to one class are saved in this folder as png. There is a csv file associated to each dataset that map a filename (with unique ID) to its associated factors of variations. Examples of dataloaders are available at `https://github.com/facebookresearch/PUG`.

D.2.1. Datasheet

Motivation	
For what purpose was the dataset created?	The 4 datasets we presented in this paper were created for representation learning research. PUG: Animals is a strong dataset for OOD research as well as for being able to better probe the representation of vision models. PUG: ImageNet was designed as an additional benchmark for ImageNet pretrained model to offer a better comprehension of vision models capabilities in term of robustness to specific factor of variations. PUG: SPAR showcase how synthetic data can be used to evaluate VLMs understanding while PUG: AR4T can be leveraged to fine-tune them.
Who created the dataset and on behalf of which entity?	This dataset was created by the FAIR team at Meta AI.
Who funded the creation of the dataset?	Meta.
Composition	

What do the instances that comprise the dataset represent?	The instances represent images of animals in various environment for PUG: Animals. In contrast PUG: ImageNet contains 151 object classes (the full list is available in Appendix D.2.3). PUG: SPAR uses the sames assets as PUG: Animal while PUG: AR4T use the same objects as PUG: ImageNet.
How many instances are there in total?	**PUG: Animals**: 215 040 images; see Appendix D.2.2. **PUG: ImageNet**: 88,328 images; see Appendix D.2.3. **PUG: SPAR**: 43,560 images; see Appendix D.2.4. **PUG: AR4T**: 249,986 images for training and 23,216 test images; see Appendix D.2.5.
Does the dataset contain all possible instances or is it a sample (not necessarily random) of instances from a larger set?	PUG: Animals contains all possible combination of factors of variations. In contrast PUG: ImageNet was sampled by changing only 1 factor at a time and is therefore a random sample of the distribution. Images in PUG: SPAR were sampled using all possible combination of factors of variations (with the exception that for the attributes the blue or grass animal is always on the left). Image-text pairs in PUG: AR4T were randomly sampled.
What data does each instance consist of?	For PUG: Animals, PUG: ImageNet, PUG: SPAR we release images along the factor of variation. For PUG: SPAR, we release the script to generate the captions from the factors of variations. For PUG: AR4T, we release images along with corresponding captions.

Is there a label or target associated with each instance?	Yes, a csv file. Each instance have a row in this csv files with all the factors of variation used to generate this image. For PUG: Animals, the csv file contain the following columns: *filename, world_name, character_name, character_scale, camera_yaw, character_texture* while for PUG: ImageNet, it contains: *filename, world_name, character_name, character_label, character_rotation_yaw, character_rotation_roll, character_rotation_pitch, character_scale, camera_roll, camera_pitch, camera_yaw, character_texture, scene_light.* For PUG: SPAR, the csv contains: *filename, world_name, character_name, character2_name, character1_pos, character2_pos, character_texture, character2_texture* For PUG: AR4T, the csv contains: *Relation, Actor1Category, Actor2Category, Actor1Name, Actor2Name, Actor1Location, Actor2Location, Actor1Rotation, Actor2Rotation, Actor1Scale, Actor2Scale, Actor1Texture, Actor2Texture, Actor1Attribute, Actor2Attribute, Camera_roll, Camera_pitch, Camera_yaw, caption, alt_caption, Level, World.Name, filename, filename_neg, filepath*

Is any information missing from individual instances?	No, all relevant information is included.
Are relationships between individual instances made explicit?	N/A.
Are there recommended data splits?	There is no specific split concerning PUG: Animals because this dataset should be used for OOD research. We primarily let the researchers choose their own held out or training/validation/testing split to train their models. In contrast, PUG:ImageNet and PUG:SPAR should only be used as an additional test set. For PUG: AR4T, splits are described in Appendix D.2.5.
Are there any errors, sources of noise, or redundancies in the dataset?	We did not explicitly filter for occlusion, so some images may contain occluded views. PUG: Animals and PUG:SPAR are very clean and each animal is easily identifiable. In contrast, PUG: ImageNet and PUG: AR4T leverage assets from Sketchfab and the asset quality vary significantly.
Is the dataset self-contained, or does it link to or otherwise rely on external resources?	The dataset is self-contained however the assets that were used to build the dataset belongs to external sources which are listed in the github at `https://github.com/facebookresearch/PUG`.
Does the dataset contain data that might be considered confidential?	No.
Does the dataset contain data that, if viewed directly, might be offensive, insulting, threatening, or might otherwise cause anxiety?	No.
Collection	

How was the data associated with each instance acquired?	The data (3D assets) were acquired through the Unreal Engine Marketplace `https://www.unrealengine.com/marketplace/en-US/store` and Sketchfab `https://sketchfab.com/`. Assets were then incorporated into the Unreal Engine to generate realistic 3D scenes and corresponding images. The 3D assets were manually selected to ensure high quality.
What mechanisms or procedures were used to collect the data?	Manual human curation. Assets were manually collected.
If the dataset is a sample from a larger set, what was the sampling strategy?	For PUG: Animals and PUG: SPAR, all combinations are included. For PUG: ImageNet and PUG: AR4T, a random sample of possible combinations is provided.
Who was involved in the data collection process and how were they compensated?	Only the authors of this work were involved.
Over what timeframe was the data collected?	The data were collected between June 2022 and June 2023
Were any ethical review processes conducted?	No.
Did you collect the data from the individuals in question directly, or obtain it via third parties or other sources (e.g., websites)?	Third parties: Unreal Engine Marketplace `https://www.unrealengine.com/marketplace/en-US/store` and Sketchfab `https://sketchfab.com/`.

Were the individuals in question notified about the data collection? If so, please describe (or show with screenshots or other information) how notice was provided, and provide a link or other access point to, or otherwise reproduce, the exact language of the notification itself.	There is no personally identifiable information in our datasets as they are purely synthetic and contain no images of people. We purchased 3D assets from different marketplaces where required, however we did not explicitly contact the individual creators.
Did the individuals in question consent to the collection and use of their data? If so, please describe (or show with screenshots or other information) how consent was requested and provided, and provide a link or other access point to, or otherwise reproduce, the exact language to which the individuals consented.	N/A. See above.
If consent was obtained, were the consenting individuals provided with a mechanism to revoke their consent in the future or for certain uses? If so, please provide a description, as well as a link or other access point to the mechanism (if appropriate).	N/A. See above.

Has an analysis of the potential impact of the dataset and its use on data subjects (e.g., a data protection impact analysis) been conducted? If so, please provide a description of this analysis, including the outcomes, as well as a link or other access point to any supporting documentation.	No data about specific individuals is included in these data. See above.

Preprocessing

Was any preprocessing/cleaning/labeling of the data done?	N/A.
Was the "raw" data saved in addition to the preprocessed/-cleaned/labeled data?	N/A.
Is the software that was used to preprocess/clean/label the data available?	N/A.

Uses

Has the dataset been used for any tasks already?	Yes, these data were used for the experiments that were presented in this paper.
Is there a repository that links to any or all papers or systems that use the dataset?	No.
What (other) tasks could the dataset be used for?	These Datasets could be used widely for evaluating and training neural networks. For example, assessing disentanglement of models with respect to PUG: Animals factors of variation (e.g. with DCI metric Eastwood and Williams [71]).

Is there anything about the composition of the dataset or the way it was collected and preprocessed/cleaned/labeled that might impact future uses?	No.
Are there tasks for which the dataset should not be used?	These datasets **should not be used** for generative modelling purposes.
Distribution	
Will the dataset be distributed to third parties outside of the entity on behalf of which the dataset was created?	Yes, the dataset will be publicly distributed.
How will the dataset will be distributed?	Tarball on a website.
Will the dataset be distributed under a copyright or other intellectual property (IP) license, and/or under applicable terms of use (ToU)?	The license of the dataset is **cc-by-nc with the mention that these data should not be used for generative AI purposes**.
Have any third parties imposed IP-based or other restrictions on the data associated with the instances?	See EpicGames [72].
Do any export controls or other regulatory restrictions apply to the dataset or to individual instances?	N/A
Maintenance	
Who will be supporting/hosting/maintaining the dataset?	Meta AI.

How can the owner/curator/-manager of the dataset be contacted?	Please contact the corresponding author of this paper.
Is there an erratum?	No.
Will the dataset be updated?	Yes the dataset will be updated with versioning.
If the dataset relates to people, are there applicable limits on the retention of the data associated with the instances (e.g., were the individuals in question told that their data would be retained for a fixed period of time and then deleted)? If so, please describe these limits and explain how they will be enforced.	N/A.
Will older versions of the dataset continue to be supported/hosted/maintained? If so, please describe how. If not, please describe how its obsolescence will be communicated to dataset consumers.	It depends. If the dataset is updated because one of the asset creators has requested to remove their assets, we will not continue to host the dataset containing these assets. Only the newer version of the dataset which will not contain these assets will be available.
If others want to extend/augment/build on/contribute to the dataset, is there a mechanism for them to do so?	No mechanisms are in place yet, but they can contact the authors of this paper if they would like to contribute.

Table 1. Datasheet for PUG, following the framework introduced by Gebru et al. [82].

D.2.2. PUG: Animals

PUG Animals contains 215 040 pre-rendered images using 70 animals assets, 64 backgrounds, 3 sizes, 4 textures, under 4 different camera orientations. To create PUG: Animals,

we use Animals assets from the following bundle in the Epic Game Marketplace (`https://www.unrealengine.com/marketplace/en-US/product/complete-animals/reviews`). The list of environments used can be found in the dataset folder or at `https://github.com/facebookresearch/PUG`. Below, we list all the values for the factors of variation, we have used:

- **World_Name** : ["Egypt", "Desert", "AmusementPark", "ArcadeClub", "Arena", "Battleground", "Catacombs", "Tableland", "EuropeanStreet", "JunkYard", "OceanFloor", "Racetrack", "Ruins", "SciFiCity", "SciFiGarage", "SpaceIsland", "SpaceHangar", "SpatialStation", "TokyoDay", "TokyoNight", "TrainStation", "Bridge", "Beach", "BusStationInterior", "BusStationExterior", "Subway", "IndoorStairs", "Bar", "ScreeningCheckpoint", "Circus", "Appartment", "Hallway", "TrashRoom", "FuturisticSubway", "Footbridge", "BoxingRing", "Hangar", "Mansion", "ShoppingMall", "ConferenceRoom", "SpacePort", "VillageOutskirt","VillageSquare","Courtyard", "ElvenRuins", "Forge", "Library", "Museum", "Gallery", "ModernGallery", "Opera", "AncientAgora", "Restaurant", "RuralAustralia", "AustralianRoad", "ShadyRoad", "SaltFlats", "Castle", "StylizedEgypt", "Temple", "Snow", "Grass", "DryGrass", "Forest"],
- **Character_Name** : ["Goldfish","Caribou","Elephant","Camel","Penguin","Cassowary","Zebra", "Turtle","Bear","Beaver","Capybara","Crocodile","Armadillo","Cat","Gecko","Crow","GiantAnteater", "GiantTortoise","KomodoDragon","Rhinoceros","Dolphin","EarlessSeal","FruitBat","Goat", "Hippopotamus","Horse","Impala","Lion","Orca","Pig","Rabbit","Squirrel", "Tapir","Wildbeest","Wolf", "Anylosaurus","BlackRockFish","Parasaurolophus","PoisonDartFrog","Spinosaurus","Triceraptos","Chicken", "HarpyEagle","Ostrich","Raven","RedCrownedCrane","Robin","Seagull","Secretarybird","Shoebill","Swan", "Toucan","Vulture","Ammonite","Ant","Scorpion","GoldBeetle","Hornet","SnowCrab","Tarantula","WhiteShark", "Tuna","Arowana", "Ayu", "Betta", "Koi", "Pirarucu", "Salmon", "Cattle", "Jerboa"],
- **Character_Scale** : [0.7, 1.0, 1.3],
- **Camera_Yaw** : [0, 45, 225, 315],
- **Character_Texture** : ["Default", "Sky", "Grass", "Asphalt"]

PUG: Animals is built by using all combinations of the factor of variation above. In Figure 1, we show random images from the PUG: animal dataset that highlight the diversity of this dataset.

Following Liu et al. [142], we present a comparison in Table 2 with other datasets that are often used in OOD research:

D.2.3. PUG: ImageNet

PUG: ImageNet contains 88,328 pre-rendered images using 724 assets representing 151 ImageNet classes with 64 backgrounds, 7 sizes, 10 textures, 18 different camera orientation,

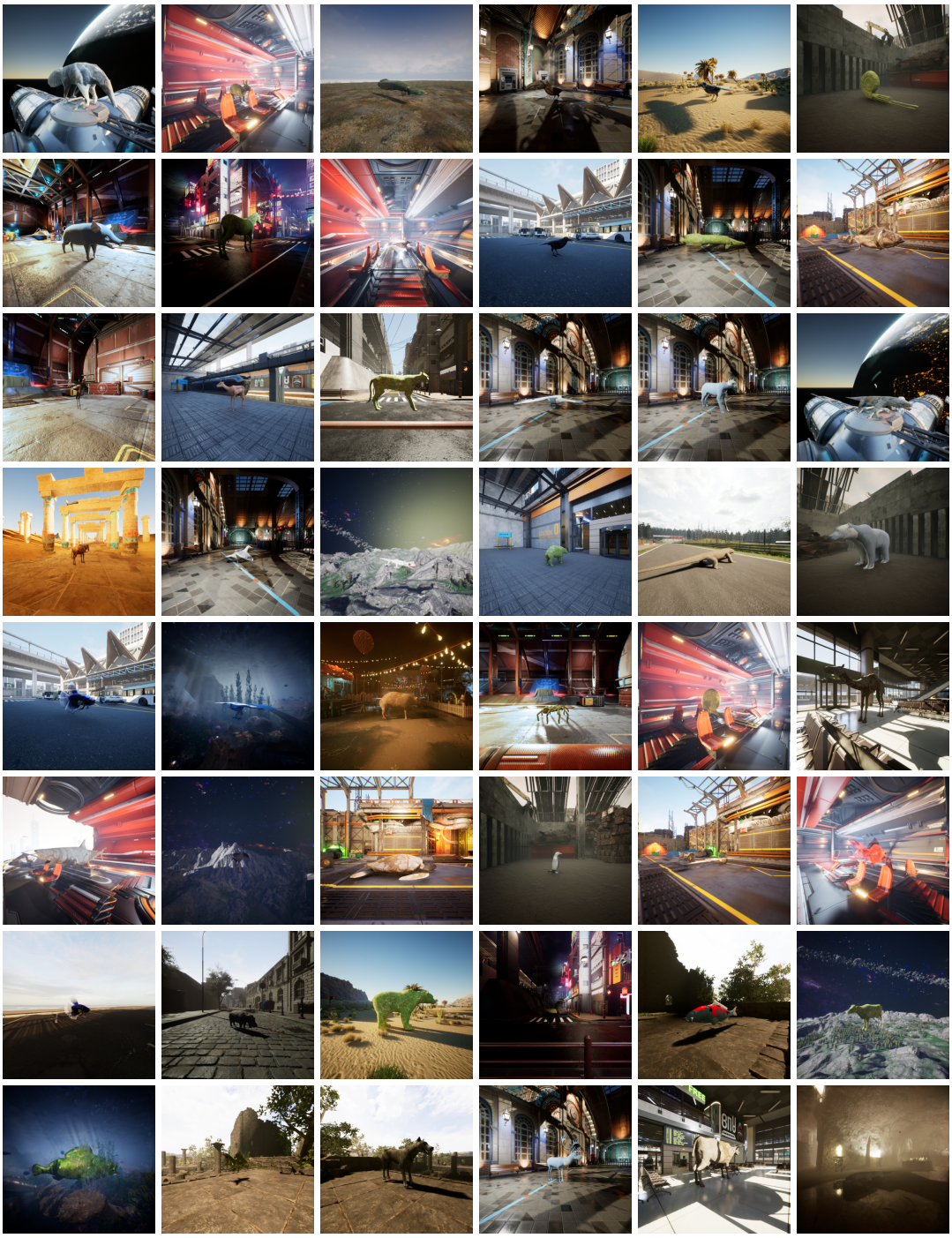


Fig. 1. Random images taken from the PUG: Animals dataset.

18 different character orientation and 7 light intensity. Below is the values we used for each of these factors:

Image Data	**Colored MNIST** [7]	**MNIST-R** [83]	**Waterbirds** [176]	**Biased-Cars** [148]	**Nico++** [228]	**PUG: Animals** Ours
# Domains	3	6	2	-	10	64
# Categories	2	10	2	5	80	70
# Examples	-	6k	4.8k	450k	230k	215k
Shift Type	Color	Angle	Background	Views	Background	Back./Text./Size./View
Image Type	Digits	Digits	Birds	Synthetic Cars	Real Objects	Synthetic Animals

Table 2. Comparing PUG: Animals with other datasets traditionally used for OOD research. In contrast to other datasets that have variations across only a single domain, that have noisy annotations or that are to unrealistic, PUG: Animal over high quality images with reliable annotations across different domains such as the background,texture,size and view.

- **World_Name** : ["Egypt", "Desert", "AmusementPark", "ArcadeClub", "Arena", "Battleground", "Catacombs", "Tableland", "EuropeanStreet", "JunkYard", "OceanFloor", "Racetrack", "Ruins", "SciFiCity", "SciFiGarage", "SpaceIsland", "SpaceHangar", "SpatialStation", "TokyoDay", "TokyoNight", "TrainStation", "Bridge", "Beach", "BusStationInterior", "BusStationExterior", "Subway", "IndoorStairs", "Bar", "ScreeningCheckpoint", "Circus", "Appartment", "Hallway", "TrashRoom", "FuturisticSubway", "Footbridge", "BoxingRing", "Hangar", "Mansion", "ShoppingMall", "ConferenceRoom", "SpacePort", "VillageOutskirt","VillageSquare","Courtyard", "ElvenRuins", "Forge", "Library", "Museum", "Gallery", "ModernGallery", "Opera", "AncientAgora", "Restaurant", "RuralAustralia", "AustralianRoad", "ShadyRoad", "SaltFlats", "Castle", "StylizedEgypt", "Temple", "Snow", "Grass", "DryGrass", "Forest"],
- **Character_Name** : 724 Sketchfab assets (See github for the list)
- **Character_Rotation_Yaw** : [0, 45, 135, 180, 225, 270],
- **Character_Rotation_Roll** : [45, 90, 135, 180, 225, 270],
- **Character_Rotation_Pitch** : [45, 90, 135, 180, 225, 270],
- **Character_Scale** : [0.5, 0.6, 0.7, 0.8, 1.3, 1.6],
- **Camera_Roll** : [45, 90, 135, 180, 225, 270],
- **Camera_Pitch** : [240, 260, 280, 300, 320, 340],
- **Camera_Yaw** : [0, 45, 135, 180, 225, 270],
- **Character_Texture** : ["Default", "Sky", "Green", "Gray", "Red", "Grass", "Color", "Black", "Curtain"],,
- **Scene_Light** : ["255,255,255,0","0,0,255,0", "0,255,0,0", "255,0,0,0", "0,255,255,0", "255,0,255,0", "255,255,0,0"] (The value for the lights are in RGBA format)

To generate PUG:ImageNet, we change only one factor at a time for each assets. When changing the background (World_Name), all the other factors (Camera/Object Orientation, Size, Texture, Light) are at 0 or at their default value. When changing the other factors (Camera/Object Orientation, Size, Texture), the background is set to "SaltFlats_0" (Which is the most *basic* background). When changing the light of the scene, we have used the environment "Opera".

The assets in PUG:ImageNet were selected base on 151 ImageNet classes which are listed below:

['BirdHouse', 'Chest', 'Bagel', 'WarPlane', 'Rocking_Chair', 'Bridge', 'Street_Sign', 'Cabbage', 'Pay_Phone', 'Butternut_Squash', 'JellyFish', 'Jack_O_Lantern', 'Bookcase', 'Stonewall', 'Punching_Bag', 'Toaster', 'Mushroom', 'Frog', 'Jeep', 'Television', 'Pineapple', 'Vacuum', 'Torch', 'Carousel', 'Desk', 'WineBottle', 'Wallet', 'Dining_Table', 'Military_uniform', 'Car_Wheel', 'Table_Lamb', 'Digital_Watch', 'Electric_Fan', 'Sweatshirt', 'Komodo_dragon', 'Racket', 'Cheeseburger', 'Can_Opener', 'Pomegranate', 'Convertible', 'Laptop', 'Chicken_hen', 'Wolf', 'Bulletproof_vest', 'Shield', 'Bathtub', 'Throne', 'Lighter', 'Bycicle', 'Cofee_Mug', 'Motor_Scooter', 'Jean', 'Soccer_Ball', 'Vending_machine', 'Hatchet', 'Umbrella', 'Bear', 'Artichoke', 'Vase', 'Radiator', 'SpaceShuttle', 'Manhole_Cover', 'Polaroid_Camera', 'Traffic_Light', 'Radio', 'Soup_Bowl', 'Zucchini', 'Barrel', 'Tennis_Ball', 'Sunglasses', 'Microwave', 'Joystick', 'Aircraft_Carrier', 'Fox', 'Submarine', 'BasketBall', 'Running_Shoe', 'Chain-saw', 'Piano', 'Crate', 'Loupe', 'Minivan', 'Shirt', 'Remote_controler', 'Airliner', 'Sock', 'Shovel', 'Mask', 'Tractor', 'Sandal', 'Wooden_Spoon', 'Drum', 'Goldfish', 'Gasmask', 'Mailbox', 'Volley_Ball', 'Banana', 'Penguin', 'Sliding_Door', 'Pool_Table', 'Burrito', 'Candle', 'Purse', 'Canoe', 'Typewriter_Keyboard', 'Espresso_maker', 'Carton', 'Park_Bench', 'Screen', 'African_crocodile', 'Cat', 'Hay', 'Elephant', 'WaterBottle', 'Modem', 'Palace', 'Ice_Cream', 'Washer', 'Sewing_Machine', 'HairDryer', 'Rabbit', 'Dishwasher', 'Bell_Pepper', 'Ambulance', 'French_Loaf', 'Refrigerator', 'Mouse', 'Obelisk', 'Starfish', 'Brocolli', 'Microphone', 'Great_white_shark', 'Power-drill', 'Locomotive', 'Perfume', 'Whale', 'Screwdriver', 'Dial_telephone', 'Backpack', 'Harmonica', 'Binocular', 'Skirt', 'Pizza', 'Cowboy_Hat', 'Computer_Keyboard', 'Kangarou', 'Baseball', 'Tile_Roof', 'Lawn_Mower', 'Safe', 'Cellular_telephone']

In Figure 2, we show random images taken from the PUG: ImageNet dataset.

D.2.4. PUG: SPAR

PUG: SPAR contains 43,560 pre-rendered images using 32 animals assets, 10 backgrounds, 4 positions and 4 textures. In contrast with the other PUG datsets, PUG: SPAR contain up to two animal in a single scene. For generating the PUG: SPAR dataset, we utilize the same subset of assets as PUG: Animals.

- **World_Name** : ['Desert', 'Arena', 'OceanFloor', 'Racetrack', 'TokyoDay', 'Circus', 'BoxingRing', 'AustralianRoad', 'SaltFlats', 'Museum'],
- **Character_Name** : ['Goldfish', 'Caribou', 'Elephant', 'Camel', 'Penguin', 'Zebra', 'Bear', 'Beaver', 'Cattle', 'Armadillo', 'Gecko', 'Crow', 'Scorpion', 'GiantTortoise', 'Tarantula', 'Rhinoceros', 'Dolphin', 'EarlessSeal', 'Goat', 'Hippopotamus', 'Horse', 'Impala', 'Lion', 'Orca', 'Pig', 'Rabbit', 'Squirrel', 'Chicken', 'WhiteShark', 'Anlylosaurus', 'BlackRockFish', 'PoisonDartFrog'],
- **Character_pos** : ["Left/Right", "Bottom/Top"]
- **Character_Texture** : ["Default", "Blue/Red", "Grass/Stone"]

In Figure 2, we show random images taken from the PUG: SPAR.

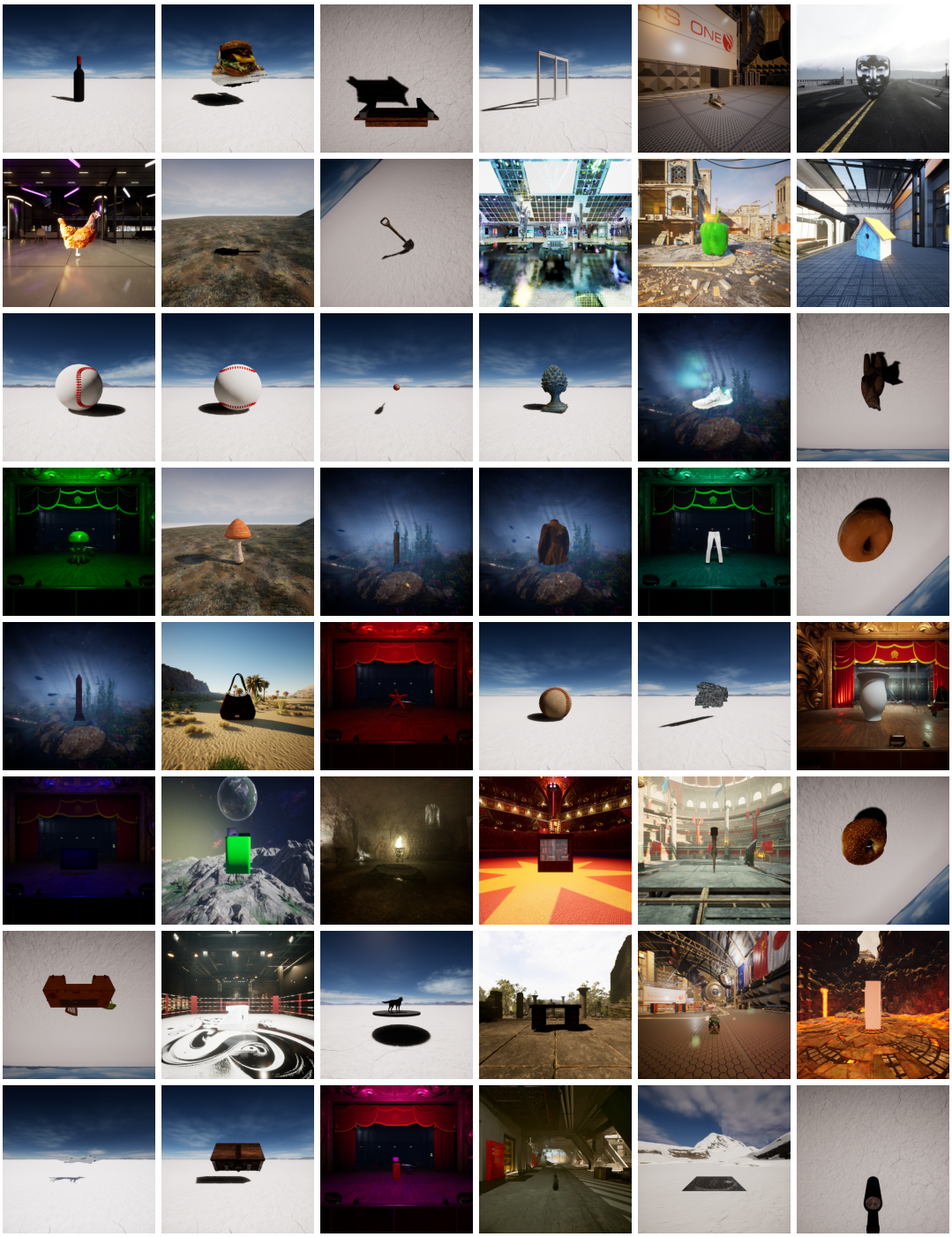


Fig. 2. Random images taken from the PUG: ImageNet dataset.

D.2.5. PUG: AR4T

Assets and Environments. For generating the PUG: AR4T dataset, we utilize the same subset of Sketchfab assets as used in PUG: ImageNet. This leaves us with a set of 680 unique assets chosen from across 151 ImageNet categories, each manually inspected to quality control

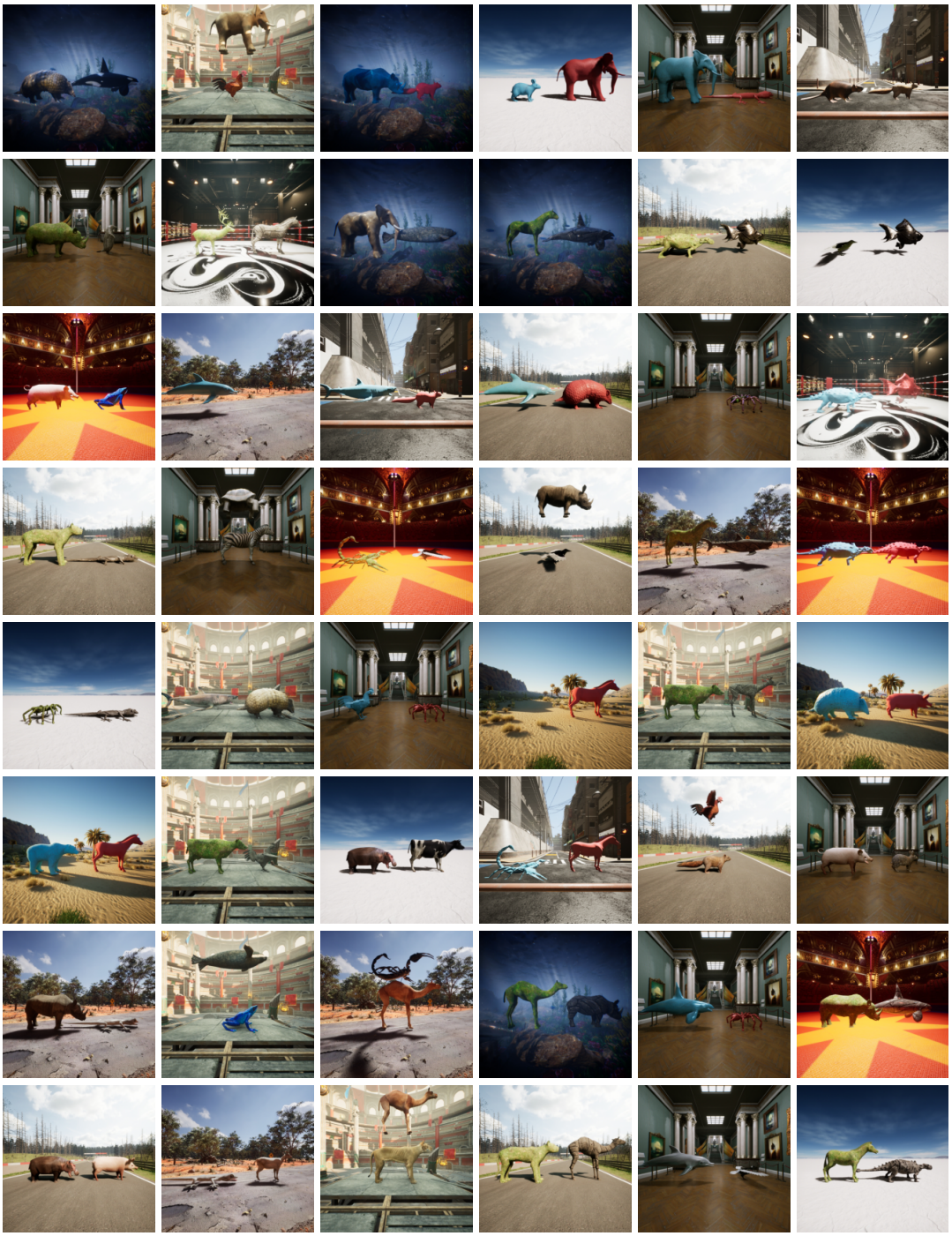


Fig. 3. Random images taken from the PUG: SPAR dataset.

for photo-realism as much as possible. For this PUG dataset we are primarily concerned with object-object and object-attribute information, hence we utilize a single camera and character orientation. Since Sketchfab assets differs widely in their scales, we first scaled down the longest edge of the asset bounding box to 150 pixels to normalize the order of magnitude of the asset dimensions, before any further scaling based on attributes. Next, we

select a total of 26 unique environments as our background environments. The richness of some environments enables us to manually select different camera views in each environment as a novel environment view, and we generate a total of 28 visually unique backgrounds for our PUG: AR4T dataset. We provide each background with human-intelligible descriptive names for use in the dataset captions (Table 3). The PUG: AR4T dataset is composed with two subset described in the following sections.

Environment Name (with camera view variant)	Descriptive Caption for PUG	Environment Name (with camera view variant)	Descriptive Caption for PUG
Arena	arena	IceRoad	icy road
VillageOutskirt	village	Jungle	jungle
VillageSquare	village	Library	library
Battleground	battleground	Museum	museum
Beach	beach	OceanFloor	ocean floor
Bridge	bridge	AncientAgora	castle outskirts
Circus	circus	Racetrack	race track
Cliff	cliff	RuralAustralia	rural wilderness
Courtyard	courtyard	AustralianRoad	desert road
Egypt	egypt	SaltFlats	salt flats
ElvenRuins	ruins	SpaceIsland	space
EuropeanStreet	european street	StylizedEgypt	egypt
FightingArena	fighting arena	Temple	temple
Forge	forge	TrainStation	train station

Table 3. Environments and descriptive captions used in PUG: AR4T

PUG: AR4T-Relations. For generating the PUG: AR4T-Relations subset, we utilize the spatial relationships from Visual Genome that are not symmetric, as noted in the ARO benchmark [235]. The set of relationships used in PUG: AR4T-Relations is given in Table 4. It consists of three unary relations (`at, in, inside`) and ten binary relations (`above, on, on top of, behind, in front of, below, beneath, under, to the left of, to the right of`.) For each relation, the corresponding objects are picked randomly from the set of assets, and placed in the scene based on the co-ordinates given in Table 4. We add a random offset in the range [0-25] along each dimension for every individual object for every scene, so that the dataset does not contain shortcuts where the object locations can directly inform the underlying relation. The size of each asset is chosen randomly on a scale of 1-10, where 1 corresponds to 110 pixels and 10 to 200 pixels for the longest edge of the scaled asset.

The caption for a scene is generated using one of two templates based on whether the spatial relationship in the scene is unary or binary:

- **Unary Relation:** *"[ImageNet class label of asset 1] [relation] [human-intelligible background description]"* e.g. Banana inside museum
- **Binary Relation:** *"[ImageNet class label of asset 1] [relation] [ImageNet class label of asset 2] [(random) unary relation] [human-intelligible background description]"* e.g. Banana to the left of chair inside museum

For each binary relation, we also sample the corresponding hard negative scene and caption, by replacing the relation with its negative from Table 4 such that the semantic meaning of the scene and caption represents a switch from the original relation between

Relation	Object 1 coordinates (wrt origin)	Object 2 coordinates (wrt origin)	Negative Relation
At	(0, 0, 0)	N/A	N/A
In	(0, 0, 0)	N/A	N/A
Inside	(0, 0, 0)	N/A	N/A
Above	(0, 0, 300)	(0, 0, 0)	Below, Beneath, Under
On top of	(0, 0, 300)	(0, 0, 0)	Below, Beneath, Under
On	(0, 0, 300)	(0, 0, 0)	Below, Beneath, Under
Below	(0, 0, 0)	(0, 0, 300)	Above, On Top Of, On
Beneath	(0, 0, 0)	(0, 0, 300)	Above, On Top Of, On
Under	(0, 0, 0)	(0, 0, 300)	Above, On Top Of, On
Behind	(150, 0, 0)	(-150, 0, 0)	In front of
In front of	(-150, 0, 0)	(150, 0, 0)	Behind
To the left of	(0, -150, 0)	(0, 150, 0)	To the right of
To the right of	(0, 150, 0)	(0, -150, 0)	To the left of

Table 4. Relations, corresponding asset locations wrt to camera origin, and corresponding hard negative relation used in PUG: AR4T

objects. For example, the negative of *"Banana to the left of chair inside museum"* is given by *"Banana to the right of chair inside museum"*.

We sample a total of 310 binary relation scenes (155 random + 155 negatives), and 310 unary relation scenes for each background, thus leading to a dataset of 112,840 image-caption samples (28 backgrounds x (310 pairs x 10 binary relations + 310 pairs x 3 unary relations)). The PUG:Relations dataset generation process described above is summarized as psueduocode in Algorithm 1.

Lastly, we split the dataset into 101,920 train and 10,920 test image-caption samples, such that the test set is balanced by background and relations (28 backgrounds x 13 relations x 30 samples). We also release the subset of training and test samples that only contain pairs of objects in scenes along with their hard negatives, which contains 78,400 training and 8,400 test samples, or 39,200 training pairs and 4,200 test pairs. This subset enables Winoground [198] style evaluation as well as training with hard visual negative mining for VLMs in future work. The PUG:Relations dataset generation process described above is summarized as psueduocode in Algorithm 1

PUG: AR4T-Attributes. For PUG: AR4T-Attributes the selection of assets, relations between assets, and spatial locations of assets is done exactly as for PUG: AR4T-Relations. However, since the focus of this dataset is on the object-attribute pairs, the relations between objects are not represented in the corresponding scene caption in any form. The attribute for each object is chosen randomly from the set of 53 attributes described in Table 5. For size based attributes, the asset's material instance remains the same but its size is varied between 50 pixels (`short, small, little, tiny`) and 200 pixels (`big, long, tall, large`). For all

Attribute Category	Attribute	Variants	Attribute Category	Attribute	Variants
Material	Brick	3	Color	Black	2
	Metal	2		Blue	2
	Wood	2		Yellow	2
	Glass	2		White	2
	Cloth	2		Green	2
	Plastic	2		Gray	2
	Rock	2		Brown	2
Size	Big	1		Red	2
	Long	1		Silver	2
	Tall	1		Orange	2
	Large	1		Pink	2
	Short	1		Gold	2
	Small	1	Texture	Striped	2
	Little	1		Dark	2
	Tiny	1		Cloudy	2
				Total=	53

Table 5. Attributes and corresponding number of variants used in PUG: AR4T-Attributes

other attributes, the material instance of the object is changed in Unreal using Pytorch Multiverse.

The caption of the scene is generated using one of two templates based on whether the spatial relationship in the scene is unary or binary:

- **Unary Relation:** *"[Attribute of asset 1] [ImageNet class label of asset 1] [relation] [human-intelligible background description]"* e.g. Green banana inside museum
- **Binary Relation:** *"[Attribute of asset 1][ImageNet class label of asset 1] and [Attribute of asset 2][ImageNet class label of asset 2] [(random) unary relation] [human-intelligible background description]"* e.g. Green banana and large chair inside museum

For each binary relation, we also sample the corresponding hard negative scene and caption, by swapping the attributes between objects such that the semantic meaning of the scene and caption represents a switch from the original object-attribute associations. For example, the negative of *"Green banana and large chair inside museum"* is given by *"Large banana and green chair inside museum"*. For each of the 53 object attributes, we sample 2 scenes of it in conjugation with another object and attribute, and 2 scenes of the attribute in isolation in a scene. We repeat this for each possible background, thus leading to a dataset of 160,272 image-caption samples (28 backgrounds x (53 attributes x (53 attributes x 2 samples) + 2 samples)). The PUG:Attributes dataset generation process described above is summarized as pseudocode in Algorithm 2.

Lastly, we split the dataset into 147,976 train and 12,296 test images. For each attribute pair in a scene, we select 4 image-caption samples in the test set (4 samples x 53 attributes x 53 attributes = 11,236 sample). And we sample the remaining test samples by selecting 20 image-caption samples for each attribute in isolation in a scene (20 samples x 53 attributes =

1,060 samples). Similar to PUG-attributes, we also separately release the subset of training and test set with scenes and captions containing only pairs of scenes with their corresponding hard negatives, leading to 146,158 training samples and 11,236 test samples, or 73,0739 training and 5,618 test sample pairs.

Caption variants in PUG: AR4T. In order to emphasize the idea that each scene can have multiple descriptive captions associated with it, we utilize a simple template to generate alternate captions for scenes with binary relations that are semantically consistent but linguistically different. During fine-tuning, the model randomly sees either the original caption or the alternate caption. The presence of alternate captions also prevents the VLM to learn shortcuts between the position of the object descriptions in captions and the underlying spatial relationship or object-attribute association.

- **PUG: Relations** *"[ImageNet class label of asset 2] [negative relation] [ImageNet class label of asset 1] [(random) unary relation] [human-intelligible background description]"* e.g. The alternate caption for 'Banana to the left of chair inside museum' is 'Chair to the right of banana inside museum'
- **PUG: Attributes** *"[Attribute of asset 2][ImageNet class label of asset 2] and [Attribute of asset 1][ImageNet class label of asset 1] [(random) unary relation] [human-intelligible background description]"* e.g. The alternate caption for 'Green banana and large chair inside museum' is 'Large chair and green banana inside museum'

Algorithm 1 PUG: AR4T-Relations subset generation

```
Unary = {At, In, Inside}
Categories = {Set of ImageNet Class Labels}
Environments = {Set of Unreal Environments}
Relations = {Set of Relations}
NegRelations = {Dictionary of relations as key, semantically negative relations as values}
Assets = {Dictionary of Sketchfab assets, keys being ImageNet Class Labels}
AssetLocations = {Function that returns locations of assets based on relation with a random offset}
Dataset = φ
for env in Env do
    for rel in Rel do
        ▷ For binary relations we also add the negative scene+caption to the dataset, thus the effective number of samples per background is 2*15 = 30
        if rel ∈ Unary then
            num_samples = 15
        else
            num_samples = 30
        end if
        for i = 1 to num_samples do
            cat1 = random.choice(Categories)
            asset1 = random.choice(Assets[cat1])
            if rel ∉ Unary then
                cat2 = random.choice(Categories)
                asset2 = random.choice(Assets[cat2])
            end if
            location1, location2 = AssetLocations(rel)
            scene1 = TorchMultiverse(asset1, asset2, location1, location2, env)
            if rel ∉ Unary then
                rel2 = random.choice(Unary)
                caption1 = cat1 + ' ' + rel + ' ' + cat2 + ' ' + rel2 + ' ' + env
                Dataset ∪ {scene1, caption1}
                ▷ Generate hard negative scene and caption
                scene1 = TorchMultiverse(asset1, asset2, location2, location1, env)
                caption2 = cat1 + ' ' + NegRelations[rel] + ' ' + cat2 + ' ' + rel2 + ' ' + env
                Dataset ∪ {scene2, caption2}
            else
                caption1 = cat1 + ' ' + rel + ' ' + env
                Dataset ∪ {scene1, caption1}
            end if
        end for
    end for
end for
return Dataset
```

Algorithm 2 PUG: AR4T-Attributes subset generation

```
Unary = {At, In, Inside}
Categories = {Set of ImageNet Class Labels}
Environments = {Set of Unreal Environments}
Relations = {Set of Relations}
Attributes = {Set of Attributes}
Assets = {Dictionary of Sketchfab assets, keys being ImageNet Class Labels}
AssetLocations = {Function that returns locations of assets based on relation with a random offset}
Dataset = φ
for att1 in Attributes do
    for att2 in Attributes + [None] do
        if att2 == None then
            num_samples = 20
        else
            ▷ For binary relations we also add the negative scene+caption to the dataset, thus the effective number of samples per background and attribute is 2*2 = 4
            num_samples = 2
        end if
        for i = 1 to num_samples do env = random.choice(Environments)
            if att2 == None then
                rel = random.choice(Unary)
            else
                rel = random.choice(Relations - Unary)
            end if
            cat1 = random.choice(Categories)
            asset1 = random.choice(Assets[cat1])
            if att2 != None then
                cat2 = random.choice(Categories)
                asset2 = random.choice(Assets[cat2])
            end if
            location1, location2 = AssetLocations(rel)
            scene1 = TorchMultiverse(asset1, asset2, location1, location2, att1, att2, env)
            if rel ∉ Unary then
                rel2 = random.choice(Unary)
                caption1 = att1 + ' ' + cat1 + ' and ' + att2 + ' ' + cat2 + ' ' + rel2 + ' ' + env
                Dataset ∪ {scene1, caption1}
                ▷ Generate hard negative scene and caption by switching object-attribute associations
                scene2 = TorchMultiverse(asset1, asset2, location1, location2, att2, att1, env)
                caption2 = att2 + ' ' + cat1 + ' and ' + att1 + ' ' + cat2 + ' ' + rel2 + ' ' + env
                Dataset ∪ {scene2, caption2}
            else
                caption1 = att1 + ' ' + cat1 + ' ' + rel + ' ' + env
                Dataset ∪ {scene1, caption1}
            end if
        end for
    end for
end for
return Dataset
```

D.3. Additional experimental details

To demonstrate the usefulness of PUG: Animals for OOD research, we trained a classifier on held-out factors split on PUG: Animals and evaluate its generalization on the held-out factors of variations, In the first experiment, we held out some factors of variation during training (backgrounds, sizes or textures) except for a specific number of animals C and use the held-out data as validation set. Thus, $C = 0$ means that the network never saw the factor during training (this is an OOD scenario with unseen factors) while $C = 10$ implies that the network saw this factor for at least 10 different animals (OOD scenario with unseen combinations of factors). In Figure 4, we present our results training a ResNet50 with different held-out factors. Every model reached more than 99% accuracy on the training set. First, we trained on 50 backgrounds and used the remaining 14 backgrounds for validation: here, the network reached an accuracy of 80%. However, when using only 30 backgrounds for training and using the remaining 34 as validation, the accuracy dropped significantly. Interestingly, showing every background for some of the animals (having unseen combinations of factors instead of just unseen factors) decreases performance. In contrast, for texture, we found that having at least 10 animals for which every possible textures are seen during training improves generalization. Interestingly, the network overfits much more to the grass texture relative to the default one. Lastly, when looking at the size factor, it seems that training on medium size assets leads to good generalization on small and large assets while training only on small assets leads to worse performance on medium and large assets.

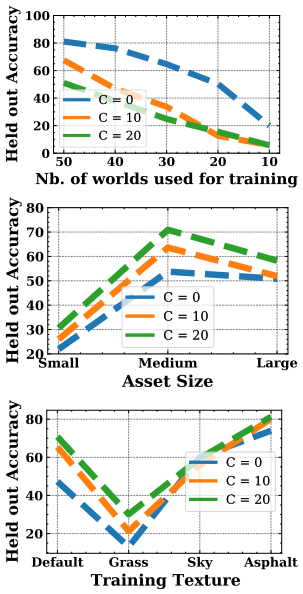


Fig. 4. Accuracy on held out factors with PUG: Animals. Each line and value C correspond to the number of animals for which all the factors are seen. The test space is built by taking all the factors minus the training factors. If we train on the Default texture, then the network is evaluated on Grass, Sky and Asphalt. If we train on 50 backgrounds, then we evaluate on 64 (total number of backgrounds) - 50 (training background) = 14 backgrounds.

D.3.1. Equivariance study details

In section 5.3.2, we used PUG: Animals to study how foundation vision-language models behave with respect to changes in factors of variations. We showed high image and text equivariance with respect to background, and text equivariance with respect to size and texture too. Here, we provide more details and results.

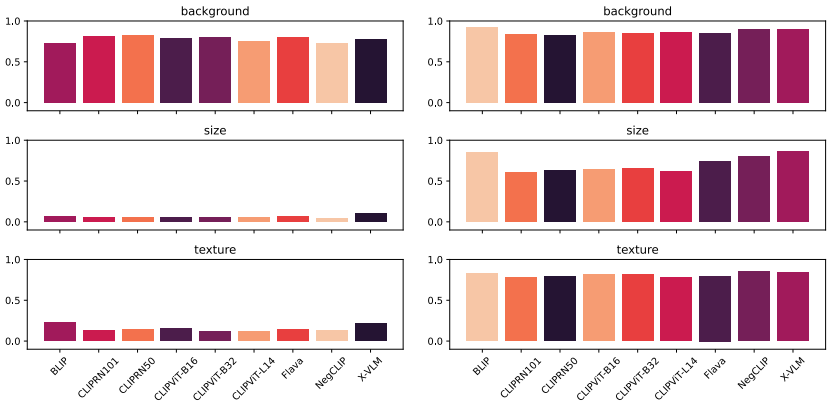
In our study, we use the following pretrained models:

- BLIP with ViT base backbone from the Huggingface transformers library [223], trained on COCO dataset [140],

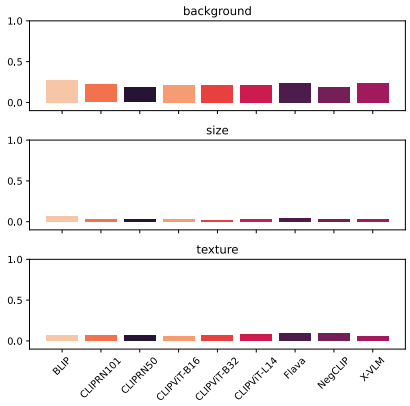
- NegCLIP from [235],
- As in [235] we use X-VLM pretrained on COCO dataset from `https://github.com/zengyan-97/X-VLM`,
- Flava with ViT-B/32 backbone from Huggingface transformers (`https://huggingface.co/facebook/flava-full`)
- CLIP models all come from OpenAI CLIP `https://github.com/openai/CLIP`. We use the versions with ResNet50, ResNet101, ViT-L/14, ViT-B/16, ViT-B/32.

We compute equivariance to each of the factors of variations. Since the text captions do not take into account camera and asset orientations, when we compute the equivariance with respect to the other factors we take only samples for a given orientation of the character and camera (0 for roll, pitch and yaw in both cases). Furthermore, in the text caption, we replace sizes by three adjectives as follows: 0.7 is mapped to small, 1.0 to medium and 1.3 to big. Inspired by [34], we compute equivariance as the alignment between embedding difference vectors. That is, we compute embedding difference vectors $z_i - z_j$ where z_i and z_j are the (normalized) embeddings of two images (or texts) of an object undergoing a given factor change. We then measure pairwise alignment as cosine similarity of embedding difference vectors (either between image pairs, text pairs, or image-text pairs) corresponding to the same factor change. We report averaged cosine similarity of randomly paired vectors, and a higher value implies higher equivariance.

Note that a model can present image equivariance but no text equivariance (caption and image are not guaranteed to be encoded in the same vector), or have high equivariance across modalities but no image or text equivariance and vice-versa. In Figure 6 we report image equivariance with respect to the orientation of the camera yaw. We see that there is little to no equivariance to it, suggesting that image embeddings are more predictable when changing the background than the camera yaw. Note that foundation model representations belong to the hypersphere, yet our measurement of equivariance as parallelism (measured with cosine similarity) relies on Euclidean geometry. Still, cosine similarity is a starting point to showcase how PUG: Animals can be used to study models' representational spaces. This could also explain the higher equivariance values of text representations: since textual captions follow the same template, embeddings might be close to each other (relative to image embedding distances). In this case the hypersphere locally behaves like an Euclidean space [133], for which the Euclidean geometry is better suited. We leave for future work the exploration more complex equivariance metrics potentially based on spherical geometry to study foundation models' representational spaces. Studying model's representations is key to better understanding and improving them Bouchacourt et al. [34], Xie et al. [226], Ushio et al. [206], Lenc and Vedaldi [135]. Our study showcases that PUG: Animals advantages (rich diversity of factors, knowledge of their values, control either



(a) Image equivariance **(b)** Text equivariance



(c) Across modalities equivariance

Fig. 5. Measuring foundation models equivariance thanks to PUG: Animals: all three factors.

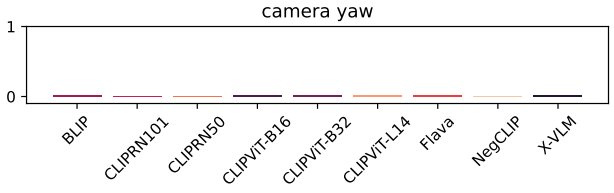


Fig. 6. Additional image equivariance results with respect to camera yaw.

one factor at a time but also all together) make it a great dataset to study state-of-the-art models representational properties.

D.3.2. Classification with held out sets

In section D.3, we study how one can leverage PUG: Animsl to study OOD generalization in two settings: 1) Generalization on unseen factors 2) Generalization on unseen combination of factors.

For this experiment, we use PUG: Animals with held out sets. Typically, we random select a number of number of animals (0, 10 or 20) within our 70 assets. Then for the remaining animals we decided to remove from PUG: Animals a number of background, object size or object texture. This give us a training set. Then the images that were excluded from this training set are put as a test or held out set in which we measure the performance of a model. This model is typically a Resnet50 trained for 100 epochs with AdamW as optimizer with a batch size of 2048.

D.3.3. Robustness of SOTA models additional details

In addition to evaluating robustness for the models in the main paper, in Table 6 we provide an analysis of several additional models including the recent self-supervised DINOv2 model as well as BLIP a contrastive vision-language model. Parenthesis indicate the pretraining dataset: ImageNet 21k, LVD-142M, JFT 300M, LAION 400M, and LAION 2B. Models without parethesis are pretrained on the standard ImageNet-1k dataset. For ResNet models we use the publicly available pretrained checkpoints from the Timm package based on the training recipe from Wightman et al. [222]. For the vision transformer models and Swin we use the pretrained models from the Timm package with patch size 16 for ViT and Swin Base with a patch size of 4 and window size of 7. For the BiT model we use the pretrained checkpoint trained on Google's JFT 300M dataset from the Timm package with a ResNetv2 101 architecture. For DINOv2, we use the officially released repo to evaluate the base ViT architecture trained on 132 million samples [162]. For BLIP we use the checkpoint available in HuggingFace and evaluate the model using zero shot classification via the prompt 'This is a photo of a [ ]'. We evaluate CLIP model variants in a similar zero shot fashion and rely on OpenCLIP's implementation. The parenthesis indicates the pretraining dataset size from the LAION dataset. We report the average accuracy as each factor (see columns of Table 6) varies.

We also measure the relationship between standard in-distribution accuracy and robustness based on the accuracy as each factor in PUG:ImageNet varies. We measure Pearson's correlation coefficient between ImageNet accuracy and accuracy for each factor in Table 7. We find no statistically significant relationship between standard classification accuracy and factor robustness.

D.3.3.1. Performances. To understand if the differences in performance between the real ImageNet and our PUG dataset is caused by a sim-to-real gap or by the factor of variations, we show below the zero-shot accuracy obtained with a pretrained resnet101 for each class in PUG: ImageNet. There is only 3 classes for which there is not a single configurations of the factors that lead to a correct classification. For all the other classes, there is always at least one configuration for which the network is correctly predicting the class. In that instance,

		PUG: ImageNet									
	ImageNet	Camera_Yaw	Camera_Pitch	Camera_Roll	Object_Yaw	Object_Pitch	Object_Roll	Object_Scale	Object_Texture	Scene_Light	Background
ResNet50	81.5	38.1	33.1	26.9	38.0	23.6	22.9	35.7	27.0	13.6	29.5
ResNet101	82.3	43.4	35.9	29.4	45.1	26.7	25.6	39.7	31.1	14.1	32.8
BiT (JFT300M)	80.3	40.5	32.3	26.0	42.1	23.6	22.8	37.3	23.4	6.3	20.5
DINOv2 (LVD-142M)	84.5	45.6	41.1	37.4	47.5	28.8	28.5	43.1	35.0	6.1	30.9
Flava (PMD 70M)	75.5	31.7	23.4	17.6	30.8	17.6	15.4	30.5	24.2	7.8	21.9
Swin	83.6	56.0	45.6	41.8	56.9	35.3	34.2	52.9	40.1	19.1	42.0
ViT-Base	84.3	37.5	34.3	31.7	38.0	21.8	20.5	33.0	28.5	4.1	26.6
ViT-Large	85.8	52.2	40.4	37.1	52.4	30.4	28.4	46.4	42.9	8.9	34.6
BLIP (100+M)	–	0.5	0.4	0.5	0.8	0.6	0.7	0.9	0.7	0.7	0.7
CLIPViTB32 (2B)	66.6	44.0	31.5	24.1	43.8	24.8	21.8	42.2	34.7	3.3	26.0
CLIPViTB32 (400M)	62.9	41.7	30.2	22.1	41.6	23.8	20.9	40.1	34.4	5.7	24.4
CLIPViTL14 (2B)	75.3	49.7	34.9	28.2	50.3	26.3	25.3	46.8	39.4	4.8	30.8
CLIPViTL14 (400M)	72.8	52.3	39.8	35.7	51.8	29.0	26.4	50.6	41.1	4.3	33.0

Table 6. Robustness measured by average accuracy across factors. We report zero shot classification accuracy for BLIP, Flava, all CLIP models. The pretraining dataset is indicated in parenthesis next to each model name with ImageNet-1k being the default unless otherwise indicated.

factor	correlation	pvalue
Object Pitch	0.29	0.45
Camera Roll	0.61	0.08
Camera Pitch	0.53	0.14
Camera Yaw	0.12	0.76
Background	0.43	0.25
Object Yaw	0.18	0.64
Object Texture	-0.10	0.81
Scene Light	0.53	0.14
Object Scale	-0.05	0.90
Object Roll	0.45	0.22

Table 7. We compute the correlation between standard ImageNet classification and robustness based on the accuracy for each factor. We find no statistically significant relationship for standard classification and factor robustness.

we assume that if the objects in a given class are correctly predicted at least one time, the failures in predicting the correct class for the same objects with different factors is probably due to the changes of factors.

Zero-shot top-1 accuracy per class Soccer_Ball: 100.00 , Pineapple: 95.62 , Barrel: 95.00 , Cellular_telephone: 89.45 , Pomegranate: 88.12 , Jack_O_Lantern: 85.94 , Vase: 85.62 , BirdHouse: 81.88 , BasketBall: 81.25 , Sewing_Machine: 80.94 , Umbrella: 80.00 , Washer: 78.75 , Pool_Table: 76.88 , Baseball: 76.88 , Safe: 76.25 , Cabbage: 75.78 , Cofee_Mug: 75.31 , Mask: 74.06 , Brocolli: 73.44 , Starfish: 72.50 , Rocking_Chair: 71.25 , Punching_Bag: 70.94 , Chicken_hen: 69.38 , WineBottle: 66.88 , Gasmask: 66.56 , Joystick: 64.06 , Television: 63.44 , Chest: 63.44 , Elephant: 62.50 , Bell_Pepper: 61.46 , Cheeseburger: 60.62 , Pay_Phone: 60.00 , Tennis_Ball: 58.44 , Jean: 56.25 , Binocular: 55.86 , Racket: 55.62 , Motor_Scooter: 55.00 , Hay: 54.06 , Park_Bench: 53.44 , Bookcase: 53.44 , Zucchini: 52.08 , Banana: 50.00 , Sliding_Door: 48.75 , Military_uniform: 47.50 , Ambulance: 47.50 , Pizza: 47.19 , Tractor: 46.88 , Dishwasher: 46.88 , Cowboy_Hat: 46.35 , Drum: 45.31 , Typewriter_Keyboard: 44.92 , Toaster: 44.69 , Obelisk: 44.38 , Laptop: 44.38 , Throne: 43.75 , Backpack: 43.44 , Shield: 41.88 , Artichoke: 41.80 , Penguin: 41.56 , Bathtub: 40.31 , WaterBottle: 40.00 , SpaceShuttle: 37.19 , Bagel: 36.88 , Bear: 36.25 ,

Vacuum: 35.94 , Radiator: 35.62 , Shovel: 35.55 , Refrigerator: 35.00 , Running_Shoe: 34.38 , Goldfish: 34.38 , Crate: 34.38 , Polaroid_Camera: 33.98 , Table_Lamb: 33.75 , Bulletproof_vest: 33.75 , Microphone: 33.12 , Traffic_Light: 32.50 , Carton: 31.25 , Volley_Ball: 30.62 , Vending_machine: 30.62 , Lawn_Mower: 29.38 , Car_Wheel: 29.38 , Harmonica: 28.12 , Lighter: 27.50 , Carousel: 27.34 , Mailbox: 27.19 , Airliner: 27.19 , Butternut_Squash: 26.95 , Sweatshirt: 26.56 , Sock: 25.62 , French_Loaf: 25.00 , Dial_telephone: 24.61 , Rabbit: 24.06 , Remote_controler: 22.81 , Modem: 22.50 , Chain-saw: 21.35 , Screwdriver: 20.31 , Power-drill: 19.69 , Electric_Fan: 19.06 , HairDryer: 18.75 , Purse: 18.12 , Wallet: 17.50 , Sunglasses: 17.50 , Minivan: 17.50 , Cat: 15.94 , Microwave: 15.62 , Candle: 15.62 , Mushroom: 15.31 , Dining_Table: 14.06 , Ice_Cream: 13.75 , Perfume: 13.44 , Komodo_dragon: 13.44 , Bycicle: 13.12 , Wooden_Spoon: 12.81 , JellyFish: 12.81 , Canoe: 12.81 , Radio: 12.19 , Desk: 12.19 , African_crocodile: 11.88 , Hatchet: 11.25 , Sandal: 10.00 , Stonewall: 9.69 , Burrito: 9.38 , Palace: 9.06 , Mouse: 7.50 , Convertible: 7.19 , Espresso_maker: 6.88 , Can_Opener: 6.56 , Jeep: 6.25 , Fox: 6.25 , Tile_Roof: 5.86 , Street_Sign: 5.62 , WarPlane: 5.47 , Frog: 5.47 , Wolf: 5.31 , Whale: 5.00 , Torch: 5.00 , Soup_Bowl: 4.69 , Great_white_shark: 4.38 , Kangarou: 3.91 , Digital_Watch: 2.81 , Skirt: 2.50 , Computer_Keyboard: 2.50 , Piano: 1.88 , Manhole_Cover: 1.88 , Bridge: 0.94 , Aircraft_Carrier: 0.39 , Screen: 0.31 , Locomotive: 0.31 , Submarine: 0.00 , Shirt: 0.00 , Loupe: 0.00

D.3.4. Additional PUG:SPAR experiments

Instead of using all the background environments presented in the main part of the paper, we also introduce a much simple setup in which we have a single background (thus we do not need the background information in the caption anymore). The background that we choose is the simplest one named "salt flats". This is also the background for which the retrieval accuracy is the highest across all the backgrounds. In table 8, we present the performances of several VLMs when using this single environment. We can observe a significant boost in accuracy for the single object detection task for which the best model achieve 94% accuracy (this value is to contrast with the 78% accuracy obtained across all the background). This show that VLMs are definitively not robust to background changes. However as in the previous case, when probing for the positional information, the performance of the model is still decreasing significantly. We also illustrate in Figure 7 very simple failure cases on the best model.

D.3.5. CLIP fine-tuning details

We utilize the OpenCLIP framework [112] for all our CLIP experiments. The ViT-B/32 model is used as the image encoder for all our experiments. The CLIP model we fine-tune is the OpenAI 400M pre-trained model *('ViT-B-32', 'openai')*[1]. Fine-tuning on PUG: AR4T is done for 10 epochs on the 200K dataset while training is done for 2 epochs on the 1M dataset.

[1]NOTE: We do not perform any training on the proprietary 400M dataset from OpenAI. We strictly only use the pre-trained models released, and fine-tune them on our PUG datasets.

Caption	Texture	ViT-B-32 (OpenAI CLIP)	ViT-B-32 (OpenClip 2B)	ViT-L-14 (OpenAI CLIP)	ViT-L-14 (OpenClip 2B)	ViT-H-14 (OpenCLIP 2B)	ViT-G-14 (OpenCLIP 2B)	Flava	BLIP	XVLM	NegCLIP
"A photo of a [character]"	Default	57.81	75.78	91.41	88.28	94.53	**94.53**	64.06	78.91	67.19	64.84
	Blue/Red	48.44	54.69	71.88	70.31	75.00	**84.38**	42.19	53.12	48.44	48.44
	Grass/Stone	39.06	45.31	67.19	68.75	76.56	**78.12**	39.06	56.25	45.31	50.00
"A photo of a [character] on the (left/right) of the picture"	Default	29.69	37.50	39.06	40.62	50.00	**51.56**	32.81	42.19	34.38	31.25
"A photo of a [character] on the (bottom/top) of the picture"	Default	29.69	23.44	45.31	43.75	**46.88**	45.31	23.44	34.38	34.38	23.44
"A photo of a [character] and a [character]"	Default	24.56	32.47	68.85	59.67	72.66	**80.96**	18.55	40.58	24.71	21.04
	Blue/Red	10.84	18.55	38.57	30.66	34.28	**51.56**	9.86	10.94	3.12	8.01
	Grass/Stone	9.38	18.16	31.35	30.47	30.18	**47.85**	6.84	11.33	5.96	8.98
"A photo of a [character] on the left and a [character] on the right"	Default	14.11	14.92	38.10	30.85	40.83	**42.14**	13.71	23.99	14.01	14.21
"A photo of a [character] on the bottom and a [character] on the top"	Default	12.00	15.42	34.07	35.48	44.05	**45.26**	10.99	26.51	14.62	8.17
"A photo of a [character] textured with [texture1] and a [character] textured with [texture2]"	Blue/Red	4.44	6.65	19.15	15.52	20.56	**29.84**	6.15	10.28	5.44	4.13
	Grass/Stone	3.43	5.44	15.73	17.24	19.05	**27.32**	5.54	7.76	6.35	4.03

Table 8. Setup and zero-shot evaluation of CLIP models on PUG: SPAR with **caption retrieval in a single environment**. In contrast with the figure presented in the main paper, we present the result only using the salt flats environment. The motivation for this experiment is to showcase the failures mode of VLMs in a very simple setup in which the model robustness to background does not impact the prediction.

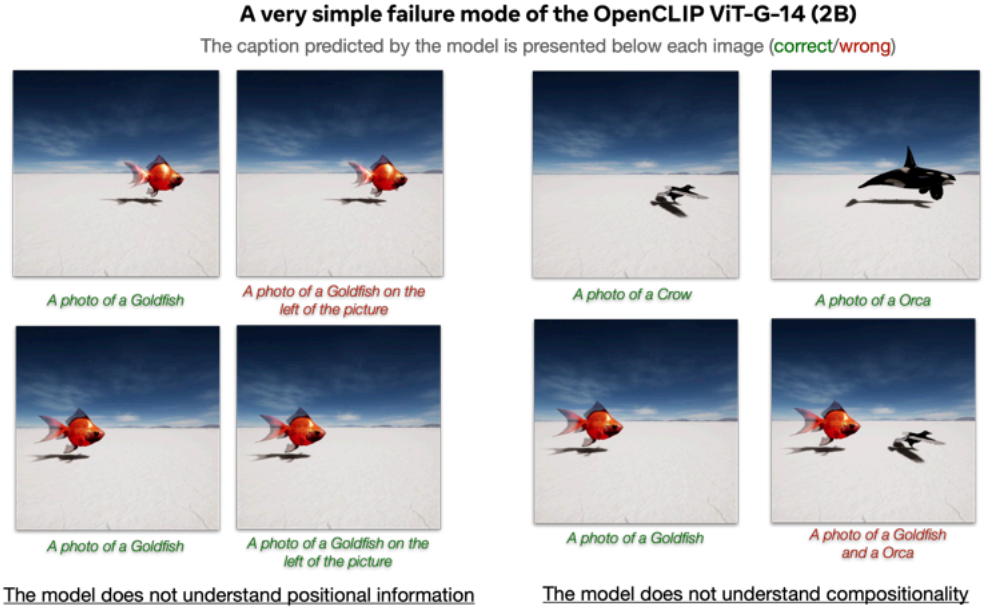


Fig. 7. Failures mode of a OpenCLIP ViT-G-14. Our PUG: SPAR dataset provides very simple images and captions and yet even large models are failing on them.